DEVELOPMENT AND USAGE OF SELF-ORGANISING MAPS IN HIGH ENERGY PHYSICS ANALYSIS WITH HIGH PERFORMANCE COMPUTING

MOHD ADLI BIN MD ALI

THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DOCTOR OF PHILOSOPHY

DEPARTMENT OF PHYSICS
FACULTY OF SCIENCE
UNIVERSITY OF MALAYA
KUALA LUMPUR

2017

UNIVERSITY OF MALAYA
ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: MOHD ADLI BIN MD ALI

Registration/Matric No: SHC120046

Name of Degree: DOCTOR OF PHILOSOPHY (EXCEPT MATHEMATICS & SCIENCE PHILOSOPHY)

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"): Development and Usage of Self-Organizing Map in High Energy Physics Analysis with High Performance Computing.

Field of Study: EXPERIMENTAL PHYSICS

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation:

ABSTRACT

The Self-Organizing Map (SOM) was put forward by Teuvo Kohonen in 1982 as a computational technique to produce a set of globally ordered quantized vectors. At the present time, it is regarded as one of the primary machine learning techniques to perform unsupervised clustering analysis on a large variety of huge data. Implementation wise, the algorithm is also parallelizable to a large extent thus allowing it to scale up/down vertically and horizontally and its adaptable to the high-performance computing environment. Thus, development of an SOM algorithm for high energy physics datasets was performed. In this research, the effects of several SOM hyperparameters such as the similarity functions, learning rate functions and map size on the clustering outcome was also performed. Moreover, a test case on how the Kullback-Leibler divergence and Multivariate Bhattacharyya Distance equation can be used as a validation parameter for SOM is performed. Additionally, it is demonstrated that a classification model can be created by staking the SOM model with a Linear Discrimination Analysis model, and the performance of this model is compared with other classification models. A demonstration of unsupervised clustering of particle physics datasets with SOM and SOM+Dirichelet Gaussian Mixture Modelling was also carried out in this research

ABSTRAK

Algoritma Petaan Swaorganisasi (SOM) telah dikemukakan oleh Teuvo Kohonen pada tahun 1982 sebagai teknik pengkomputeran bagi menghasilkan set rangkaian vektor yang tersusun secara global. Pada masa kini, teknik ini telah pun dianggap sebagai salah satu teknik utama bagi analisis kelompok tanpa pengawasan, terutama sekali bagi set data yang mengandungi bilangan entri yang tinggi dan pelbagai. Dari segi implementasinya, ia dapat dikomputasikan secara selari pada skala tinggi, sekali gus membolehkan ia dilakukan pada skala besar menggunakan komputer berprestasi tinggi. Oleh itu, satu aplikasi SOM untuk menganalisis set dat daripada fizik tenaga tinggi telah dibangunkan di dalam kajian ini. Kajian ini juga meneliti kesan beberapa hiperparameter SOM seperti fungsi persamaan, fungsi kadar pembelajaran dan saiz peta terhadap gugusan sampel yang dihasilkan. Selain itu, penulisan ini juga menunjukkan cara bagaimana persamaan Perbezaan Kullback-Leibler dan Jarak Multivariasi Bhattacharyya boleh digunakan sebagai ujian-pengesahan terhadap output yang dihasilkan oleh SOM. Penulisan ini juga menunjukkan suatu modal pengklasifikasian boleh dihasilkan dengan menggandingkan SOM bersama teknik *Linear Discrimination Analysis* (LDA), dan prestasi pengklasifikasian ini apabila dibandingkan dengan teknik pengklasifikasian yang lain. Kajian ini juga menilai kebolehan SOM+*Dirichelet Gaussian Mixture Modelling* dalam mengelompokkan data fizik tenaga tinggi yang tidak mempunyai label.

ACKNOWLEDGEMENTS

To my god, I say the verses that come out from the lips of your prophet;

'I am content with Allah as my Rubb, with Muhammad as my Messenger and with Islam as my Deen'

(Muslim)

An immense gratitude is also should be given to my mother, Pn. Rosnah Sharif, whom with her single hand raise me up till I can successfully write my Ph.D. thesis.

A special acknowledgment also should be given to Prof Wan Ahmad Tajuddin, the main supervisor of this research. He also acts as the head of National Centre for Particle Physics (NCPP) and without his effort, the centre would not even existed. The acknowledgment is also extended to the rest of NCPP member, it's visiting lecture, research students, and support staffs.

To Luis Eduardo Medina Medrano, the verse 'Tú eres el jardín de mi corazón' is written for you.

A sincere gratitude and thanks are given to Marius Cornelis van Woerden, Aidan Randle-Conde, Khoi Nguyen Nguyen, and Matt Behlmann, who all have become dearest closet friends to me.

This thesis will also be incomplete without the guidance provided by Marco Mascheroni, the Big Boss of CRAB3 Development group and a mentor in the world of Python coding. The same acknowledgment is also extended to Justas Balcas, Andres Jorge Tanasijczuk and the rest of CRAB3 team.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

2HDM	type-II two-Higgs-Doublet
Acc	Accuracy
ANN	Artificial neural network
APD	avalanche photodiodes
API	Application Programming Interface
ATLAS	A Toroidal LHC ApparatuS
AUC	Area Under Curve
BDT	Boosted Decision Tree
CE	Computing Element
CERN	Centre for European Nuclear Research
CMS	Compact Muon Solenoid
CMSSW	CMS Software
CRAB	CMS Remote Analysis Builder
CSC	cathode strips chambers
DAS	Data Aggregation System
DBS	CMS Dataset Bookkeeping System
DPGMM	Dirichlet Process Gaussian Mixture Modelling
DT	drift tubes
DT	decision tree
ECAL	electromagnetic calorimeter
Fn	False Negative
Fp	False Positive
GIC	Global Instance Cluster
GPGPU	general-purpose graphic processing unit
HB	HCAl Barrel
HCAL	Hadron Calorimeter
HE	HCAL Endcap
HF	HCAL Forward
HO	HCAL Outer
HPC	high-performance computing
I/O	input/output
IS	Information System

JSON	JavaScript Object Notation
KL-Distance	Multivariate Kullback-Leibler distance
LDA	Linear Discrimination Analysis
LFN	Logical File Name
LHC	Large Hadron Collider
LIC	Local Instance Cluster
MB-Distance	Multivariate Bhattacharyya Distance
MET	missing transverse energy magnitude
ML	Machine Learning
MSSM	Minimal Supersymmetry Standard Model
N.No	Noise Instance Number
NCPP	National Centre for Particle Physics
PFN	Physical File Name
PhEDEx	Physic Experiment Data Export
QCD	Quantum Chromo Dynamic
QDA	Quadratic Discrimination Analysis
RF	Random Forest
ROC	Receiver Operating Characteristic
RPC	resistive plate chamber
S.No.	Signal Instance Number
SE	Storage Element
SiteDB	Site Database
SM	Standard Model
SOM	Self-Organising Map
SRM	Storage Resource Manager
STD	Standard Deviation
SUSY	Supersymmetry
SVM	Support Vector Machine
TEC	Tracker EndCap
TIB	Tracker Inner Barrel
TID	Tracker Inner Disk
Tn	True Negative
TOB	Tracker Outer Barrel
Tp	True Positive
UI	User Interface

UM	Universiti Malaya
WLCG	Worldwide LHC Computing Grid
WMAgent	Workload Management Agent
Pt	Momentum

CHAPTER 1
THESIS INTRODUCTION

"*"Necessity is the mother of invention"* is a silly proverb. *"Necessity is the mother of futile dodges"* is much near to the truth. The basis of the growth of modern invention is science, and science is almost wholly the outgrowth of pleasurable intellectual curiosity

(Singler, 1996, P. 140)

1.1 Introduction

Particle physics study has always been seen as a field that is too theoretical and has little direct benefit to the society. This claim is not true of course as technologies such as X-ray, world wide web (www) and proton radiotherapy are all originated from particle physics research. Even in the current time, research into particle physics keeps pushing the limit in computing technology.

Each year, the Compact Muon Selenoid (CMS) detector record several petabytes worth of data, either from events directly coming out from the proton-proton collision or by computer simulation. There is also user-generated analysis data, computational-operation metadata and a staggering pile of detector calibration constants. In short, a particle detector produces and stores a massive amount of data in its day to day operation.

To analyse and interpret this large amount of data, particle physicists practice various computational methodologies, among them is Machine Learning (ML). In fact, ML techniques such as the Boltzmann machine, Fisher Analysis, and Decision Tree have always been employed in particle physics analysis, either on their own or in addition to statistical analysis.

Nonetheless, the development of ML techniques in particle physics itself has become stagnant in recent times. The method has not been abandoned, but there is a lack of interest in making it better. Questions such as: how can the ML technique be more efficient in using computing resources in a particle physics computational environment? Is there any new ML technique that can increase background suppression? This kind of question is not being answered in current research.

Other research fields, such as finance, meteorology, robotic and DNA sequencing are aggressively exploiting ML, employing the most contemporary techniques. Thus, it is only logical that with the amount of data, the particle physics field should also be aggressively striving for more advanced ML techniques.

This thesis is created to try fills the gap or, at least, revive the interest of employing different ML techniques in particle physics analysis. Therefore, it focuses more on the potential usage of ML, more specifically a technique calls the Self-Organizing Map (SOM) in particle physics analysis. Introduction is also given on the several computational practice that is currently being employ in the CMS collaboration.

1.2 Objective

The research objectives for this thesis are;

1. To understand the CMS computational-grid analysis workflow and contribute to its development
2. To develop a clustering and classification algorithm base on the Self-Organizing Map method for particle physics datasets

3. To measure the overall performance of the Self-Organizing Map method in clustering and classifying particle physics events.

1.3 Scope of Study

The content of this thesis sits at the interface between particle physics, ML and high-performance computing (HPC) since the ML algorithm is developed for particle physics datasets and requires HPC hardware to be executed promptly. Hence, a general introduction regarding these three different subjects is required.

This thesis emphasizes the applied aspect of ML technique for particle physics analysis. As such, only a minimum level of theoretical framework regarding the particle physics and ML algorithms is presented in this thesis.

This research will be using three datasets, plus a trivial dummy dataset, which is described in chapter 1. All three datasets are open access dataset so that all result obtain in this thesis are neutral and can be reproduced easily by other party.

The result obtained from the develop self-organizing map (SOM) technique will be compared to other ML algorithms such as the ensemble decision tree and support vector machine. However, the main algorithm studied for this research is SOM, thus, only the SOM result will be given a detail look.

Any algorithm, middleware or software that was developed in this study is meant to be executed in an HPC environment that uses commodity hardware and Linux base operating system. Specialized hardware such as supercomputers, including machine with

general-purpose graphical processing unit (GPGPU) are excluded, since this type of hardware is generally not openly available in the particle physics grid-computing environment. The content regarding computational grid environment is solely focused on the one employed by the CMS collaboration. The computational ecosystem of other particle physics collaborations is not discussed in the thesis.

1.4 Thesis Structure

In this thesis, the literature review is given in chapter 2, the first part of chapter 3 and the first part of chapter 4. The detail for computational software development effort is given in the second part of section 3 and the second part of chapter 4. The experimental result and discussion is given in chapter 5, 6 and 7. The content of each chapter is as follows:

Chapter 1: This chapter give a general overview of the research, presenting keywords such as machine learning, self-organizing map, high-performance computer, particle physics and the boundary of the overall thesis

Chapter 2: This chapter has two function; first it gives the necessary introduction to the physics involve in the three datasets that are being used in this research. This chapter covers topics such as the standard model, the extended standard model Higgs and Supersymmetry. The second part of the chapter consist of a general overview concerning the CMS detector and its components.

Chapter 3: This chapter combines three topics, first it provides an explanation regarding the CMS grid-computing ecosystem, such as the analysis and production workflow.

Secondly it focuses on the usage of a software called CMS remote analysis builder (CRAB) and the author's contribution in developing this application. The chapter ends with a short discussion regarding the effort that has been contributed by the author to the development of University Malaya High-Performance Computing Cluster called 'sifir'.

Chapter 4: The chapter starts by explaining some terminology that is commonly used in the study of ML. It then proceeds to provide an explanation regarding some of the well-known ML algorithms for classification which are, Random Forest, Support Vector Machine, Linear/Quadratic Discrimination Analysis (LDA/QDA) and shallow Neural Network. The chapter also gives some examples of particle physics research that uses ML algorithm. The chapter then provides the necessary discussion about the Dirichlet Gaussian Mixture Modelling (DPGMM) algorithm which is a clustering algorithm. From there, the chapter shifts its focus to the SOM algorithm and its implementation. A short discussion regarding the practicality of employing SOM on the global cloud infrastructure concludes this chapter.

Chapter 5: Each ML model has certain configuration parameters call hyperparameters which dictates the model performance. The objective of this chapter is to conduct experiments to study which hyperparameter configuration gives the optimum SOM model. In the conducted experiment, the Multivariate Bhattacharyya Distance and the Multivariate Kullback-Leibler distance equation are uses to gauge the SOM model performance. The first part of this chapter gives detail regarding the various SOM model hyperparameters while the second part of the chapter is about the experiment conducted. This is followed by a discussion of the obtained result, and the optimum SOM model configuration is given.

Chapter 6: The chapter gives detail about the conducted experiment to measure the performance of the SOM+LDA/QDA model in classifying events from the Higgs dataset. The accuracy and other performance scores for the SOM+LDA/QDA model are given and compared with other ML classification algorithms. The chapter also discusses on how the SOM model can be used to determine which feature combination that gives the best separation between signal and noise in the Higgs dataset.

Chapter 7: The content of this chapter is about the experiment that was carried out in determining SOM+DPGMM model clustering capability on the SUSY dataset. It shows that the developed model is capable of clustering together similar events without having explicit information about the underlying physics that is involved. This chapter also provides a demonstration on how SOM can reveal hidden patterns in the particle physics dataset.

Chapter 8: This chapter provides the research conclusion as well as suggestions for improvements that can be done.

CHAPTER 2
INTRODUCTION TO PARTICLE PHYSICS AND THE COMPACT MUON SELENOID DETECTOR

...the finder of a new elementary particle used to be rewarded by a Nobel Prize, but such a discovery now ought to be punished by a $10,000 fine.

(Willis E. Lamb Jr. 1955)

2.1 Chapter Introduction

In 2012, both the Compact Muon Solenoid (CMS) collaboration and the A Toroidal LHC ApparatuS (ATLAS) collaboration announced the discovery of a boson particle with a mass of 125 GeV (ATLAS, 2012; CMS Collaboration, 2012a) at the Large Hadron Collider (LHC) experiment, Geneva, Switzerland. The newly discovered particle is believed to be the Higgs boson, thus marking the end of a very long search for the elusive particle, a search that has been going on for the last ~48 years.

The Higgs Boson is the last remaining particle required to complete the Standard Model (SM) of particle physics. Since the introduction of the SM by several theorists such as Abdus Salam, Sheldon Glashow, and Steven Weinberg, the model has proven to be accurate in describing the interaction between elementary particle.

The first objective of this chapter is to give a basic introduction to particle physics so that the content of a particular dataset can be understood (subchapters: 2.12 - 2.2.7). Three different datasets were used in this research, a beyond-SM Higgs dataset,

supersymmetry (SUSY) dataset and the dimuon dataset. All three datasets are public, with the first two datasets published by the Centre for Machine Learning and Intelligent Systems, University of California, Irvine, while the CMS collaboration has published the third dataset (Baldi eta., 2014; McCauley, 2014).

The second objective of this chapter is to give the required information regarding the CMS detector (subchapter 2.3 - 2.3.6). The discussion concerning the CMS detector does not have a direct relation to the objective of this research. However, it provides the context for understanding the scale of computing requirements for the CMS detector that will be discussed in chapter 2.

The author would like to reiterate that this research is concern with the applied aspects of ML algorithms in particle physics analysis. Thus, only introductory level of theoretical particle physics is provided.

2.1.1 Standard Model Particle and Forces

Table 2.1 show all the particles in the current standard model (SM) which can be further divided into fermions and bosons. Both the quarks and leptons have half-integer spin, which makes them fermion particles following Fermi-Dirac statistics. On the other hand, bosons have integer spin and follow Bose-Einstein Statistics.

Table 2.1: The Standard Model of particle physics.

Quarks			Force Carrying	
up (u)	charm (c)	top (t)	gluon (g)	
down (d)	strange (s)	bottom (b)	photon (γ)	Higgs Boson (H)
Leptons			Z (Z)	
electron (e)	muon (μ)	tau (τ)		
electron neutrino(υ_e)	muon neutrino (υ_μ)	tau neutrino (υ_τ)	W (W)	

2.1.2 The Quark

In 1969, Bjorken & Paschos suggested a model in which the proton is composed of point-like constituents as a way to interpret the result from the deep-inelastic scattering on a proton at the Stanford linear accelerator (SLAC), see Bjorken & Paschos, (1969) and Bloom et al., (1969). The idea of a point-like structure to describe the composition of the proton and neutron originates from the Quark model, introduced by Gell-Mann and Zweig in 1964, (Riordan, 1992).

The quark model states that (Griffiths, 1987);

1. All baryons are composed of three quarks, consequently all antibaryon are composed of three antiquarks
2. All mesons are composed of a quark and anti-quark.

The mass and charge of each quark are given in **Table 2.2**. It is worth noting that the quarks masses are free parameters of the SM (Uzan & Leclercq, 2008), thus they have to be determined experimentally. However, the mass value obtained also depends on which renormalization scheme is used, therefore the quark mass is a scheme- dependent value, see Olive et al., (2014).

Table 2.2: The Standard Model version of quarks and its mass, (Olive et al., 2014)

Flavour	Electrical Charge		Mass (MeV)
	Quark	Antiquark	
Up	$+\frac{2}{3}$	$-\frac{2}{3}$	$2.3^{+0.7}_{-0.5}$
Down	$-\frac{1}{3}$	$+\frac{1}{3}$	$4.8^{+0.5}_{-0.3}$
Strange	$-\frac{1}{3}$	$+\frac{1}{3}$	95 ± 5
Charm	$+\frac{2}{3}$	$-\frac{2}{3}$	1275 ± 25
Bottom	$-\frac{1}{3}$	$+\frac{1}{3}$	4180 ± 30
Top	$+\frac{2}{3}$	$-\frac{2}{3}$	173210 ± 510 (stat err) ± 710 (sys err)

2.1.3 The Lepton

Different to the quarks, the charged lepton masses can be measured directly in an experiment and the values obtained are not scheme dependent (Cahn & Goldhaber, 2001).Table **2.3** show all known lepton and their properties;

Table 2.3: Mass of lepton (Olive et al., 2014)

Name	Mass (MeV)
Electron	$0.51099 \pm 11 \times 10^{-9}$
Muon	$105.65837 \pm 3.5 \times 10^{-6}$
Tau	1776.82 ± 0.16
Electron-Neutrino	$< 2 \times 10^{-6}$
Muon-Neutrino	<0.19
Tau- Electron	< 18.2

2.1.4 Standard Model Forces and the Higgs Boson

The SM contains three fundamental forces, which are the electromagnetic force, the weak force and the strong force, listed in **Table 2.4**. The gravitational force that is postulated to be carried by a particle called the graviton is excluded from the standard model, due to its formalism being based on general relativity, while SM is based on quantum field theory. Combining these two frameworks has been the primary effort in the field of quantum gravity and related fields.

Table 2.4, Table 2.4 shows the strength of each force (taken from Griffiths, (1987)). However, the value should not be taken literally as a force's strength depends upon the particle's couplings and separation. The masses and charges are taken from Olive, (2014)

Table 2.4: Force carrying particles and their masses, including the Higgs boson under the SM.

Name-Symbol	Force	Charge	Mass
Photon - γ	Electromagnetic	$< 1 \times 10^{-35}$ e	$< 1 \times 10^{-18}$ eV
W^+ / W^-	Weak	+1/-1	80.385 ± 0.015 GeV
Z	Weak	0	91.1876 ± 0.0021 GeV
Gluon - g	Strong	0	$< 6 \times 10^{-32}$ eV
Higgs	-	0	125.7 ± 0.4 GeV

2.1.5 Electromagnetic Force

The electromagnetic force governs the interactions between electrically charged particles, and it is formalized using Quantum Electrodynamics (QED). According to Griffiths (1987, p. 60), all QED interactions can be reduced to the interaction shown in **Figure 2.1 (Left)** which is an electron emitting a photon. Whereas, **Figure 2.1 (Right)** is a positron and electron annihilating each other and forming a photon. If this diagram is turned by 180°, it depicted a photon decaying into an electron-positron pair, an event that is called the 'pair production'. Another example of an electromagnetic decay is the Dalitz pair, in which a neutral pion decays into an electron, positron and photon, $\pi^0 \rightarrow e^+e^-\gamma$.

Few particles decay via the electromagnetic force and it often only happens when the decay via the strong force is forbidden. For example, the decay of the eta particle to three pions, $\eta \rightarrow \pi^+\pi^-\pi^0$ is an electromagnetic decay since it is forbidden by the selection rules of the strong force, see Perkin (1987, p. 83).

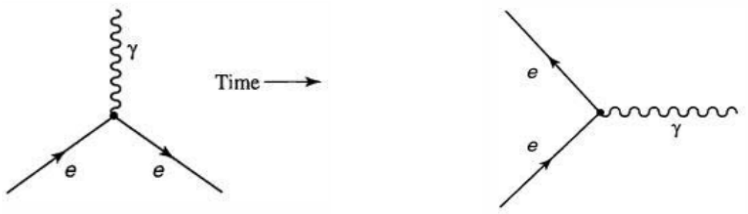


Figure 2.1 Feynman diagram of an electron emitting a photon (left) and electron – positron annihilation emitting a photon (right). Time is projected onto the x-axis with the positive time to the right. Taken from (Griffiths, 1987)

Each force has an associated coupling constant that is associated with the force strength. The coupling constant of the electromagnetic force is known as the Fine structure constant because it determines the magnitude of the spin-orbit splitting in atomic spectra, shown in equation (2.1)

$$\frac{e^2}{\hbar c} = \alpha \approx \frac{1}{137} \tag{2.1}$$

2.1.6 Weak Force

All particles experience the weak force; however, the force is much weaker than the electromagnetic and strong forces. A key signal that an interaction is mediated by the weak force is that it involves a neutrino particle. An example of this interaction is the beta decay of the neutron, $n \rightarrow p + e^- + \bar{\upsilon}_e$ and antineutrino absorption by a proton, $\bar{\upsilon}_e + p \rightarrow n + e^+$.

Particle flavour change is also a key signal of a weak interaction, for example the purely hadronic decay of Sigma-hyperon to neutron and pion, $\Sigma^- \rightarrow n + \pi^-$, in which the strange quark in the Sigma changes to a non-strange quark (Perkins, 1987).

The mediators of the weak force are the $W^{\pm}$ and Z particles and the fundamental vertices for the weak force are shown in **Figure 2.2**,

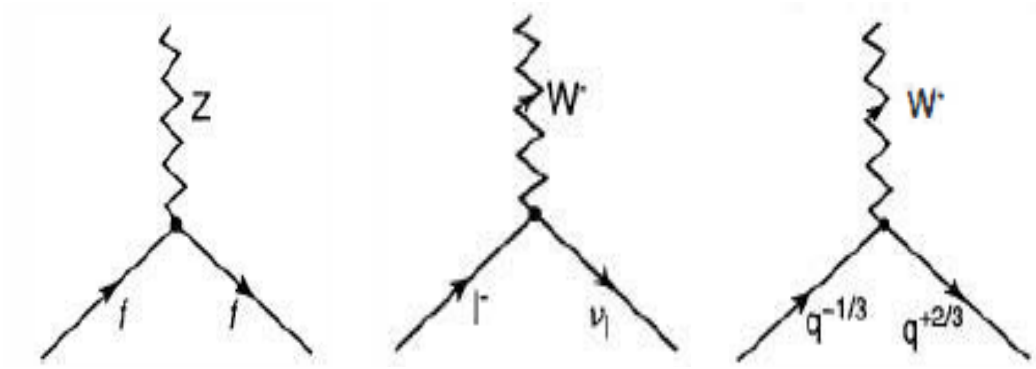


Figure 2.2: Feynman diagram of weak fundamental vertices for neutral (left) and charged (centre) leptonic interaction, and charged quark interaction (right) (Griffiths, 1987)

In 1967-1968, Glashow, Salam and Weinberg proposed that the weak coupling constant should be similar to the electromagnetic constant, thus for the weak force it will be the dimensionless constant *g* instead of *e*, equation (2.2), (Perkins, 1987). This idea leads to the unification of weak and electromagnetic forces, into the electroweak theory that in turn leads to the formalism of standard model

$$G \equiv \frac{g^2}{M_w^2} \cong 10^{-5}\ \text{GeV}^{-2} \tag{2.2}$$

2.1.7 Strong Force

The strong force is only experienced by the quarks that make up the hadron via the mediator massless particle call the gluon. However, different to the photon which only has two charge polarities (– and +, two degrees of freedom), the gluon has three colour charges and three anticolour charge (six degrees of freedom). The colour symmetry is supposed to be exact, thus, the quark-quark interaction is independent of the colour charge involved. The theoretical basis for gluon and quark interaction is called Quantum

Chromodynamics, (QCD), and the force coupling constant is given in equation (2.3) where g_s is the strong coupling.

$$\alpha_s = \frac{g_s^2}{4\pi} \cong 1 \tag{2.3}$$

An example of a strong interaction is the decay of the Sigma resonance Σ°(1385) in $K^- + p \rightarrow \Sigma^\circ(1385) \rightarrow \Lambda + \pi^\circ$. One important property of the strong force is that the colour-charge confinement; the colour charge potential, V_s between two quarks (say $Q\bar{Q}$) is usually given in the form of equation (2.4), (Perkins, 1987);

$$V_s = -\frac{4}{3}\frac{\alpha_s}{r} + kr \tag{2.4}$$

In (2.4) it can be seen that at small distance, r, the potential is small but will continue to rise with distance as the second term (of the right hand side) become more dominant. The potential will continue to increase until it is energetically more favourable to create a new pair of $Q\bar{Q}$, which explains why individual quarks are never observed.

2.2 SM Dimuon Channel

The dimuon channel ($\mu^+\mu^-$) is an interesting decay branch since it provides access to study quarkonia particles and higher order QCD corrections. Quarkonia are particles that consist of a quark and an antiquark (meson) of the same flavour, for example, J/ψ are composed of $c\bar{c}$ quark and the Upsilon meson, Υ, is composed of $b\bar{b}$ quark. Both these quarkonia can decay directly into the dimuon final state, $J/\psi \rightarrow \mu^+\mu^-$, $\Upsilon \rightarrow \mu^+\mu^-$, (D0 Collaboration, 2014; LHCb Collaborations, 2014). It is important to note here that meson that are composed of light flavour quark (up, down and strange) are not usually considered as quarkonia.

At the moment, no theoretical formalism can entirely account for the total production cross-section and spin configuration of heavy quarkonia. Estimation based on the non-relativistic QCD factorization scheme provide a first complete next-to-leading-order calculation of ϒ production, however, they overestimates production in the low transverse momentum (*pT*) region (LHCb Collaborations, 2014).

Figure 2.3 shows the distribution of invariant mass from the dimuon decay mode obtained by the CMS Collaboration, (2012b) during the 2010 data taking at a proton-proton collisions of 7 TeV. Resonance peaks belonging to the quarkonia state(s), the Z boson and other light mesons (phi and omega mesons) can be clearly seen.

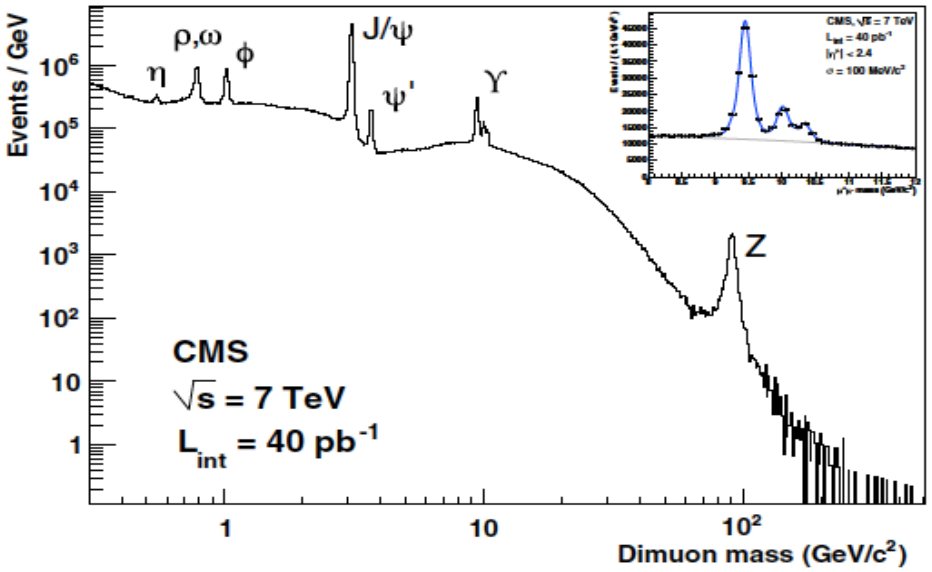


Figure 2.3: The dimuon invariant mass spectrum from the 2010 CMS collaboration data taking. The inset shows the energy between 8-12 GeV/c^2, showing the resolution of the upsilon particle.

The three upsilon ($\Upsilon(1S)$, $\Upsilon(2S)$, $\Upsilon(3S)$) mass resolution in the dimuon channel was later refined by the CMS Collaboration, (2010) for the 7 TeV data and it is shows in **Figure 2.4**.

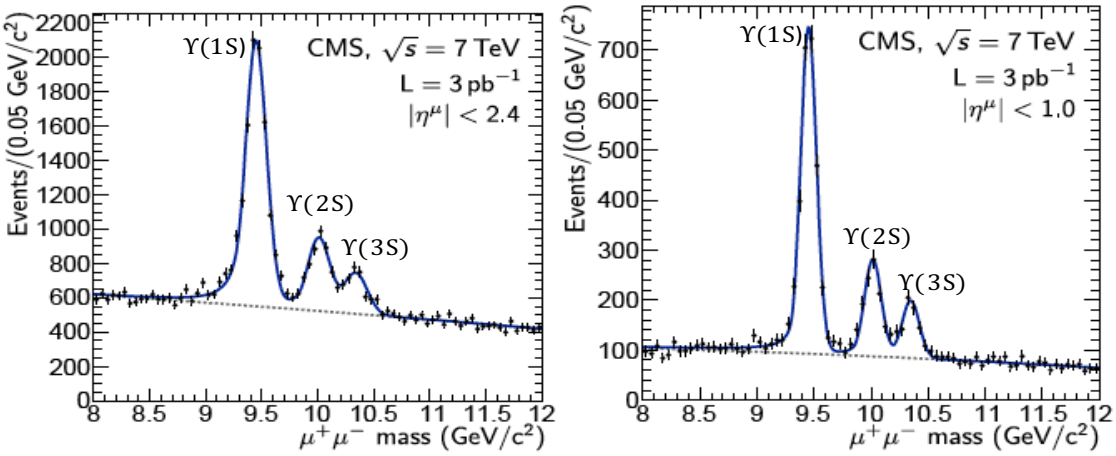


Figure 2.4: The invariant mass spectrum for the upsilon particle between two pseudorapidity ranges, $|\eta^{\mu}| < 2.4$ (right) and $|\eta^{\mu}| < 1.0$ (left). Error bars show data, the blue line shows the fit to data, and the dashed line shows the fit to background processes.

The J/ψ and its excited state $\psi(2S)$ mass resolution was later refined by the CMS Collaboration, (2011) as shown in **Figure 2.5**. With **Figure 2.4** and **Figure 2.5**, it can be stated the mass of for $\Upsilon(1S)$, $\Upsilon(2S)$, and $\Upsilon(3S)$ is 9460.30 ± 0.26 MeV, 10023.26 ± 0.31 MeV and 10355.2 ± 0.5 MeV respectively, while the mass for J/ψ and $\psi(2S)$ is 3096.916 ± 0.011 MeV and $3686.109^{+\,0.012}_{-\,0.014}$MeV, respectively, (Olive et al, 2014)

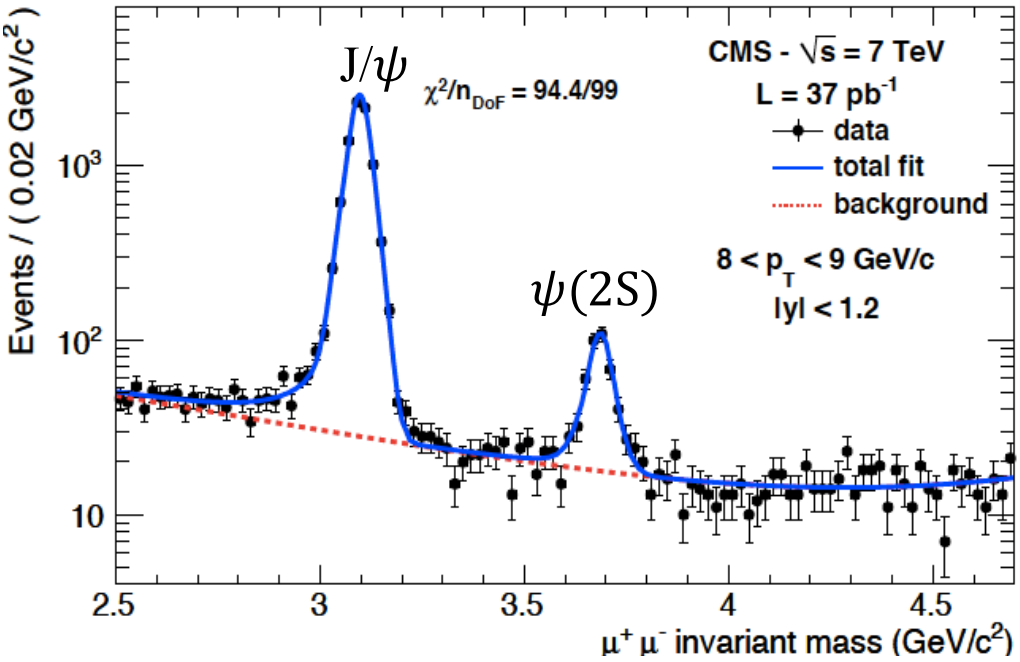


Figure 2.5: The invariant mass of J/ψ and ψ(2S) in the region of rapidity $|y| < 1.2$ and momentum $8 < p_T < 9$ GeV/c for the 7 TeV proton-proton collision at CMS Collaboration, (2011)

For a proton-proton collision, dimuon production mostly originated from the Drell-Yan process and decay of resonances. The decay of top quark to multi leptonic final states contributes to the background. In the Drell-Yan process, the quark and anti-quark from two different hadrons annihilate each other to produce a virtual photon or Z boson that decays into a dimuon pair of opposite charge, leading to the so-called dimuon continuum.

Despite the dimuon being a great channel for quarkonia and QCD formalism study, the Higgs to dimuon channel, $H \rightarrow \mu^+\mu^-$ is one of the smallest decay branching ratios for a Higgs boson with mass of 125 GeV. The ATLAS Collaboration, (2014) has stated that the branching ratio for this process is 1.5×10^{-3} at 95% confidence level, whereas the CMS Collaboration, (2015) set the upper branching limit at 1.6×10^{-3}.

2.2.1 Beyond SM Dimuon Channel

The dimuon channel also has been used to extract relevant information and set new limits regarding searches for physics beyond the SM, particularly for the Minimal Supersymmetry Standard Model (MSSM) Higgs model (CMS Collaboration, 2016; CMS Collaborations, 2012) and supersymmetry (ATLAS Collaboration, 2012). For a SM Higgs boson at the mass of 126 GeV, the upper limit of the branching ratio for $H \rightarrow \mu^+\mu^-$ is 1.5×10^{-3} according to ATLAS Collaboration, (2014). To date, no excess event beyond the SM prediction in the dimuon channel has been observed.

2.2.2 Dimuon Dataset

The CMS collaboration openly published the dimuon dataset, see McCauley, (2014). For this thesis, the total number of events used is 100,000. Even with this minute amount of data (compared to the data regularly collected by the CMS collaboration), plotting the invariant mass, **Figure 2.6,** still clearly shows the peaks of multiple resonances as seen in **Figure 2.3**, however the resonance peak for $\Upsilon(3S)$ could not be resolved from the background as shown in **Figure 2.7**;

The dimuon dataset contains the following parameters (feature) for each event;

1. The total momentum and momentum along the x, y, and z axis for both muons
2. The energy carried by both muons

3. Indication whether a given muon is a globally reconstructed muon or a tracker muon. The discussion of global and tracker muon is out of scope for this thesis, please see (CMS Collaboration, 2008) for the detail.
4. The invariant mass of the two muons

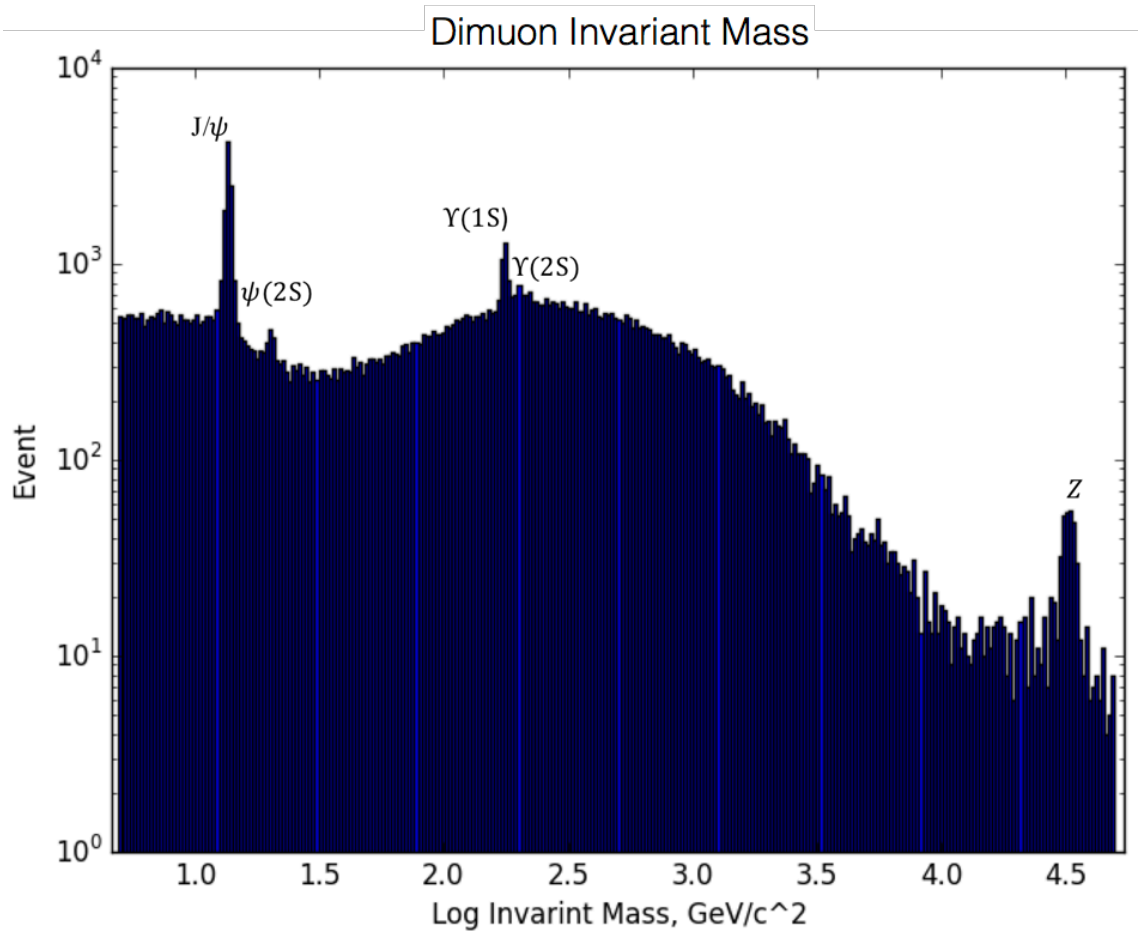


Figure 2.6: The Invariant mass spectrum obtained from the dimuon dataset that is being use for this thesis

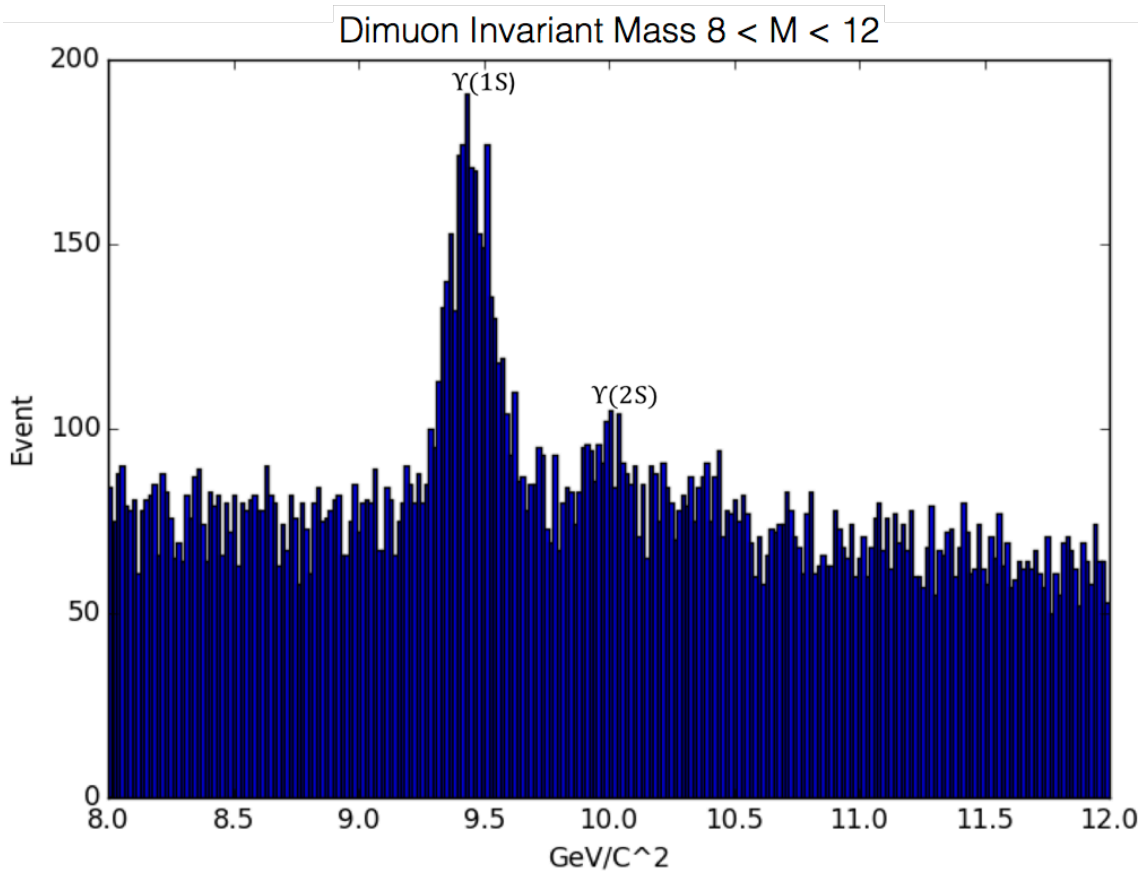


Figure 2.7: The dimuon invariant mass between 8 GeV and 12 GeV depicting the $\Upsilon(1S)$ and $\Upsilon(2S)$

2.2.3 Beyond Standard Model Higgs

It is widely believed that the standard model of the particle physics is incomplete as it fails to provide an answer to certain phenomena such as the origin of dark matter and neutrino oscillations. On top of that, the model contains several phenomenological anomalies such as the hierarchy problem (the quadratic divergence of self-energy corrections at high energy, (ATLAS Collaboration, 2013)), extent of CP violation and the exclusion of gravity (please see Griffiths, (1987) for further discussion).

Due to these weaknesses, several models such as supersymmetry (SUSY), technicolor, and extra dimensions have been proposed as extensions or alternatives to the

towards SM. In this subchapter, a short review regarding the Two-Higgs-Doublet (2HDM) is presented. The 2HDM model extends the Higgs family to additionally contain two heavy charged Higgs boson ($H^{\pm}$) and a heavier neutral state (H^0) on top of the neutral light Higgs, h^0 = 126 GeV/c. Certain 2HDM models also adds a pseudoscalar particle, *A* (ATLAS Collaboration, 2013). Both the H^0 and *A* can decay into an electroweak boson including h^0 and it is possible to detect them at the LHC if their mass is below 1 TeV(CMS Collaborations, 2014). At the time of writing this thesis, no collaborations have announced any observations of an extended Higgs sector.

2.2.4 The Higgs Dataset

In this thesis, an open access dataset containing the simulated events of cascading decay of neutral heavy Higgs, H^0 is used. The dataset was created by Baldi et al., (2014) and it is based on a case study done by ATLAS Collaboration, (2013), which in turn is based on the work of Evans et al., (2012). According to the ATLAS Collaboration, (2013) the decay mode does not follow a specific theoretical framework, but a simplified model of the (2HDM).

This dataset contains a mixture of two simulated event types that are labelled as signal and noise. Entries labelled as signal are events that originated from the process of gluon fusion that creates the $H^0$and have a final product $W^{\mp}W^{\pm}b\bar{b}$, shown in **Figure 2.8(a)**. The full decay mode is given as below;

$$gg \rightarrow H^0 \rightarrow W^{\mp}H^{\pm} \rightarrow W^{\mp}W^{\pm}h^0 \rightarrow W^{\mp}W^{\pm}b\bar{b}$$

The entries that are labelled as noise also originated from gluon fusion, but then decay into $t\bar{t}$ and a have the same final product as the signal evens, $W^{\mp}W^{\pm}b\bar{b}$, **Figure 2.8(b)**. In this datasets the masses of H^0 and $H^{\pm}$ are assumed to be $m_{H^0} = 425\ \text{GeV}$ and $m_{H^{\pm}} = 325\ \text{GeV}$.

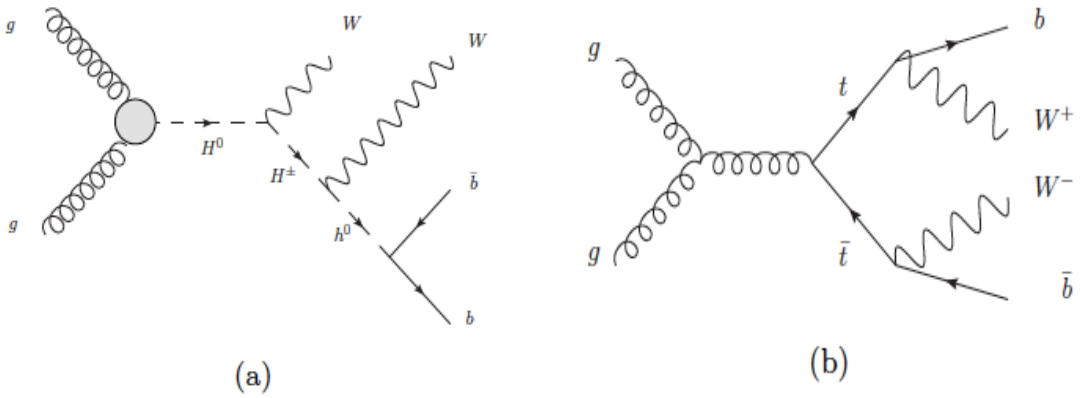


Figure 2.8: The two events simulated in the Higgs dataset.

In this dataset, each event contains the following variables;

1. The low level inputs which include;
    a. lepton energy and direction,
    b. missing transverse energy magnitude (MET),
    c. 4 B-tag jet energy and direction
2. The high-level inputs which include;
    a. the calculated mass of two jets, m_{jj} and three jets, m_{jjj}
    b. calculated mass of lepton plus missing energy, m_{lv}
    c. calculated combined mass of lepton, jet and missing energy, m_{jlv}
    d. calculated mass of two bottom quarks m_{bb}
    e. calculated combined mass of two bottom quark with a single W boson, m_{wbb} and two W boson, m_{wwbb}

The distribution of the given feature is shown in Appendix A.1

2.2.5 Supersymmetry

The foundation for supersymmetry (SUSY) models is that for every fermion there exists a boson as its counterpart and vice versa for boson, **Table 2.5.** Due to the fact that it is not known which supersymmetric particle is the lightest, it is common to call the charged and natural SUSY particles produced in an event as the chargino, $\chi^{\pm}$, and the neutralino χ^{0}.

The difficulty in the SUSY theoretical framework is that it contains a large collection of free parameters that have to be determined experimentally, thus giving various versions of SUSY. For example, the popular Minimal Supersymmetric Model (MSSM) contain 105 free parameters, which can be reduced when the method of breaking the symmetry is specified (Robichaud-Véronneau, 2013). This in turn leads to different versions of the MSSM, including;

- Gravity-mediated supersymmetry breaking (Falkowski, Lee, & Lüdeling, 2005)
- Gauge-mediated supersymmetry breaking (Giudice & Rattazzi, 1999)
- Anomaly-mediated supersymmetry breaking (Alwis, 2008)
- Gaugino-mediated supersymmetry breaking (Chacko, Luty, Nelson, & Pontón, 2000)

Nevertheless, in September of 2013, after the LHC finished its first run, Nathaniel Craig in his workshop lecture at the Galileo Galilei Institute, stated;

I do not mean this to say that there is no supersymmetry in nature. Rather, I mean that the march of null results suggests that we were mostly wrong about precisely how supersymmetry would appear at the LHC.

(Craig, 2013, p. 4)

Table 2.5: Supersymmetry particles and their Standard model partners, (Perkins, 1987)

SM – Particle	Spin	Supersymmetry – Particle	Spin
Quark, Q	$\frac{1}{2}$	Squark, $\tilde{Q}$	0
Lepton, l	$\frac{1}{2}$	Slepton, $\tilde{l}$	0
Photon, γ	1	Photino, $\tilde{\gamma}$	$\frac{1}{2}$
Gluon, G	1	Gluino, $\tilde{G}$	$\frac{1}{2}$
W	1	Wino, $\widetilde{W}$	$\frac{1}{2}$
Higgs, H	0	Shiggs, $\widetilde{H}$	$\frac{1}{2}$

Craig continued to suggest that the failure is due to the approach in building the SUSY framework that focuses on naturalness (the hierarchy problem, see Martin, (1997)) and parsimony (minimality).

2.2.6 Supersymmetry Dataset

The third dataset that is used in this thesis is called the SUSY dataset, published by Baldi et al., (2014). It contains the simulated SUSY process of: $gg \rightarrow h \rightarrow \chi^+\chi^- \rightarrow \upsilon\bar{\upsilon}\ell^+\ell^-\chi^0\chi^0$, labelled as signal, as well as the SM process: $q\bar{q} \rightarrow WW \rightarrow \upsilon\bar{\upsilon}\ell^+\ell^-$. Both of these event topologies have the same detectable final products which is a dilepton plus missing energy. The Feynman diagram for this process is given by **Figure 2.9**.

Nevertheless it is necessary to point out that Baldi et al., (2014) did not state which version of SUSY phenomenology is to be adopted as the theoretical framework for process. The mass of $\chi^{\pm}$ and χ^{0} is assumed to be $m_{\chi^{\pm}} = 200\ \text{GeV}$ and $m_{\chi^{0}} = 100\ \text{GeV}$.

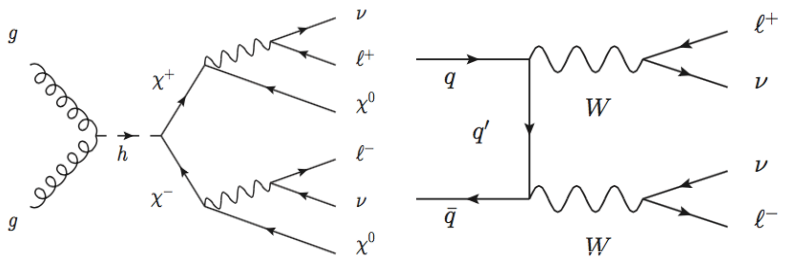


Figure 2.9: The simulated signal (Left) and noise (Right) in the SUSY dataset

To the best knowledge of the author, neither the CMS and ATLAS collaboration published any search for this particular decay channel mode. Nevertheless, the decay channel for the signal event can be considered as a neutral and oppositely charged SUSY particles decaying into an oppositely charged dilepton pair with missing energy. Hence, the most similar decay channel that has been searched by the CMS collaboration is the $pp \rightarrow \tilde{\chi}^{+}\tilde{\chi}^{-} \rightarrow \nu\tilde{\ell}\tilde{\nu}\ell \rightarrow \nu\nu\ell\ell\tilde{\chi}^{0}\tilde{\chi}^{0}$ and $pp \rightarrow \tilde{\chi}^{+}\tilde{\chi}^{-} \rightarrow \tilde{\ell}\tilde{\ell} \rightarrow \ell\ell\tilde{\chi}^{0}\tilde{\chi}^{0}$, **Figure 2.10,** (CMS Collaboration, 2014).

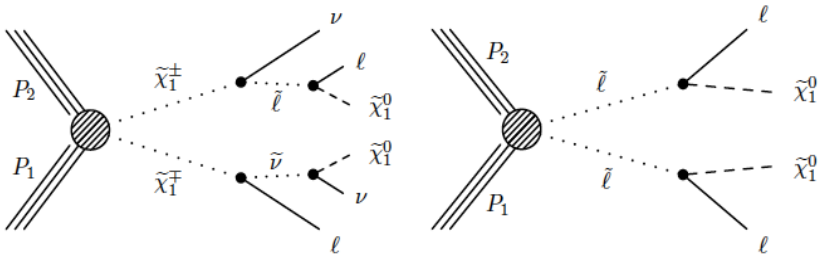


Figure 2.10: The production and decay of two chargino into two oppositely charged dileptons with missing energy by the that has been search by the CMS Collaboration, (2014)

The ATLAS Collaboration has also searched on the same channel with an additional decay mode: $pp \rightarrow \tilde{\chi}^{+}\tilde{\chi}^{-} \rightarrow \tilde{\chi}^{0}\tilde{\chi}^{0}WW \rightarrow \nu\nu\ell\ell\tilde{\chi}^{0}\tilde{\chi}^{0}$, **Figure 2.11,** (ATLAS Collaboration, 2014). At the time of writing, no deviation from the expected background SM process is observed in any of the channels.

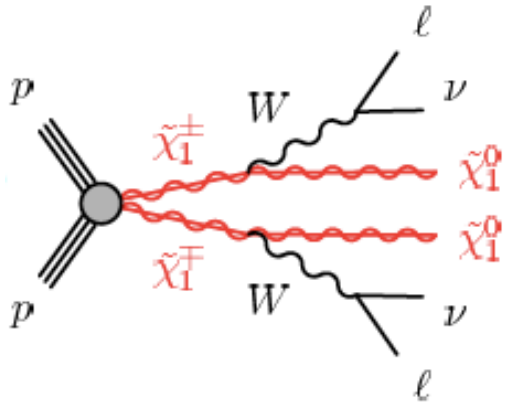


Figure 2.11: Additional decay mode of chargino to two oppositely charged leptons with missing energy that has been searched by the ATLAS Collaboration,(2014)

According to Baldi et al., (2014), each entry in the dataset has the following variable:

1. Low level features which include:
    a. Both lepton energy and direction in pseudorapidity
    b. Missing energy magnitude and direction
2. High-level features which include:
    a. Axial E_T which is the missing transvers energy along the vector defined by the charged leptons
    b. M_{T2} : Estimated mass for a particle produced in a pair and decaying in a semi-invisible manner
    c. E_T^{Rel} defined by equation (2.5) where $\Delta\emptyset$ is the minimum angle between E_T and a jet or lepton

$$E_T^{Rel} = \begin{cases} E_T \text{ if } \Delta\emptyset > \dfrac{\pi}{2} \\ E_T \sin(\Delta\emptyset) \text{ if } \Delta\emptyset < \dfrac{\pi}{2} \end{cases} \quad (2.5)$$

d. β, R and M_R are razor quantities defined in Rogan, (2010)

e. $\beta_{R+1}, \cos(\theta_{R+1}), \Delta\emptyset_R^{\beta}, M_{\Delta}^{R}, M_R^T$ and $\sqrt{\widehat{S_R}}$ are super razor quantities defined in Buckley et al., (2014)

Discussion regarding these high-level features is out of the scope of this thesis. However, the parameter M_R^T will be used later and it is given by equation (2.6);

$$(M_R^T)^2 = \frac{1}{2}\left[E_T^{miss}(q_{1T} - q_{2T}) - \boldsymbol{E_T^{miss}} \cdot (\boldsymbol{q_{1T}} - \boldsymbol{q_{2T}})\right] \quad (2.6)$$

where E_T^{miss}, q_{1T} and q_{2T} is the missing transverse energy, visible transverse moment particle 1 and the visible transverse moment particle 2 respectively. The distributions of these variables is given in Appendix A.2

2.2.7 Dataset Production

Both the Higgs and SUSY datasets are produced by computer simulation. According to the authors of the dataset, Baldi et al., (2014), MADGRAPH5 and MADGRAPH were used for the event generator for the Higgs and SUSY dataset respectively. Assuming a proton-proton collision at 8 TeV(Alwall et al.,2011). Furthermore, PYTHIA was used to simulate the showering and hardonization, while DELPHES was used for detector response (Ovyn et al., 2009; Sjöstrand et al., 2006).

2.2.8 Dataset Conclusion

This research used three different datasets with each dataset physics is governed by three different branches of particle physics. The dimuon dataset contains events recorded by the CMS muon chambers and it follow the Standard Model framework. The majority of the events registered in this dataset come from the Drell-Yan process, the decays of light mesons and the Z boson. On the other hand, the Higgs dataset contains the simulated events of a cascading decay of neutral Higgs. The simulation is based on the 2HDM model in which extends the Higgs sector to contain 2 additional heavy neutral Higgs bosons and 2 heavy charged Higgs. The third dataset is the SUSY dataset; it contains simulated events of two charginos decaying into neutralinos, a dilepton pair with opposite charge and two neutralinos. The simulated event is based on an MSSM model.

2.3 Introduction to the CMS Detector

The events in the dimuon dataset were recorded by The Compact Muon Selenoid (CMS) muon chamber. CMS is one of four large detectors situated at the LHC, located in Cessy, France. It was designed to record events from proton-proton (pp) collisions with a centre-of-mass energy = 14 TeV as well as events from lead-lead collisions at 5.5 TeV pre nucleon. This subchapter will provide a short overview of the CMS detector according to the report published by the CMS Collaboration, (2008). This thesis only uses the 2010 dimuon dataset (the other two datasets are Monte Carlo simulated data); hence, only the original design of the detector will be given and information regarding detector upgrades during the 2014 first long shutdown is not included.

2.3.1 General Overview

In terms of size, the CMS detector has a length of 21.6m, with a diameter of 14.6 m and a total weight of 12500t. The proton-proton collision point is enveloped by an all-silicon pixel-strip tracker, which in turn is surrounded by a lead-tungstate scintillating-crystal electromagnetic -calorimeter, which is also surrounded by a brass-scintillator sampling hadron calorimeter. All the inner detector is positioned inside a large-bore superconducting solenoid that also submerges the detector in a high magnetic field. The magnetic field is returned by the iron yoke which also contains four stations of muon detector. Two forward calorimeters are positioned outside the iron yoke (CMS Collaboration, (2008). A perspective view of the detector is shown in **Figure 2.12**

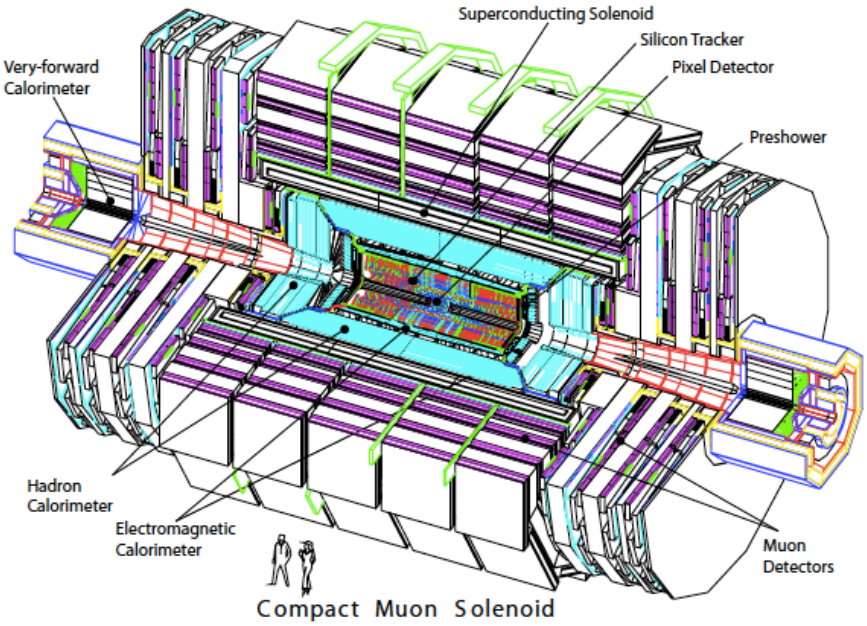


Figure 2.12: A perspective view of the CMS detector

The CMS detector was designed to meet the following objectives:

- Muons can be identified with an excellent momentum resolution in a broad range of momenta and angle. Good dimuon mass resolution and it is able to determine muon charge with little unambiguity.
- Have excellent momentum resolution and reconstruction efficiency in the inner tracker for charged particles. Have efficient triggering and online tagging for the tau, τ particle and b-jets.
- Good electromagnetic energy resolution, efficient photon and lepton isolation, and π^0 rejection wide geometric coverage, giving good mass resolution for diphoton and dielectron states.
- A large hermetic geometric coverage with fine lateral segmentation for the hadron calorimeter so that a good resolution on missing transverse energy and dijet mass can be obtained.

According to CMS Collaboration, (2014), the following coordination system is adopted for each event recorded:

- The origin point is located at the nominal collision points inside the detector.
- Y-axis points vertically upward, X-axis points towards the centre of the LHC radial beam and Z-axis lies in the counter clockwise direction of the beam line
- The polar angle θ is the angle between the positive Z-axis and XY-plane, while the azimuthal angle $\emptyset$ is measured in the XY-plane
- Pseudorapidity, η is defined by equation (2.7)

$$\eta = -ln\left(\tan\left(\frac{\theta}{2}\right)\right) \tag{2.7}$$

2.3.2 CMS Superconducting magnet

According to the CMS Collaboration, (2008), the CMS superconducting magnet is designed to sustain a 4T magnetic field during operation, storing energy up to 2.6GJ at full current, **Figure 2.13**. The flux of the magnetic field is returned by the 10,000t yoke, which is comprised of 5 wheels and 2 endcaps. The superconducting solenoid has three new features compared to the previous detector magnet:

1. Four layers of winding instead of the usual one layer (Aleph and Delphi) or a maximum of 2 layers (ZEUS and BaBar) to support the necessary ampere-turns in generating 4T of magnetic field, (ALEPH Collaboration, 1990; BaBar Collaboration, 2002; DELPHI Collaboration, 1991; ZEUS Collaboration, 1993).
2. The conductor is made from Rutherford-type cable co-extruded with pure aluminium and mechanically reinforced with aluminium alloy
3. The solenoid size is massive: 6.3m cold-bore, 12.5m in length and has a mass of 220t

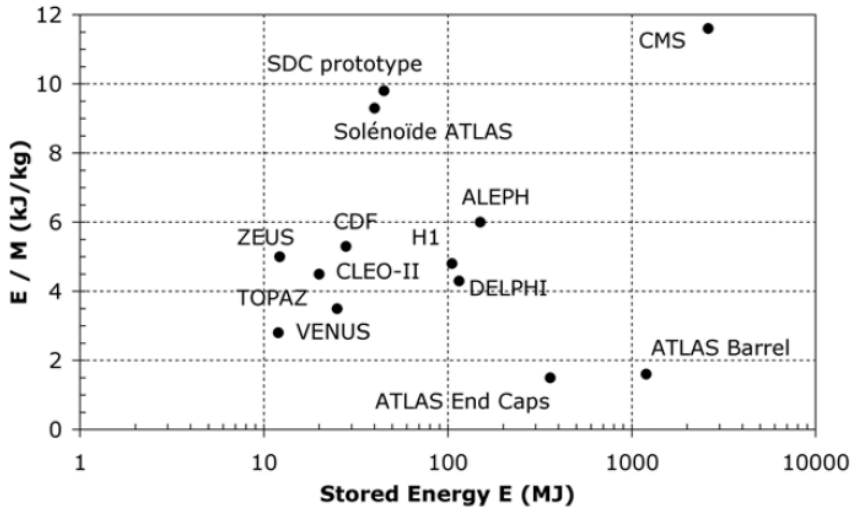


Figure 2.13: Comparison of the CMS detector magnet with other detector magnet in term of energy over mass ratio, (CMS Collaboration, 2008).

It is worth noting that even though in (CMS Collaboration, 2008) is stated that the superconducting coil is supposed to deliver a homogenous 4T magnetic field to the inner detector, subsequent publications state that the magnitude of the magnetic field deliver is only at 3.8T, (CMS Collaboration, 2010b, 2012b).

2.3.3 CMS Inner Tracker

According to (CMS Collaboration, 2008), the inner tracker has the function of providing a precise and efficient measurement of charged particle tracks emerging from LHC collisions and secondary vertices. As the LHC will produce a luminosity up to $10^{34}cm^{-2}s^{-1}$, which produce an average of 1000 particles from 20 overlapping proton-proton collisions for every 25 ns, the tracker is required to have high granularity, fast response time, be efficiently cooled and able to operate in a harsh environment with an expected lifetime of 10 years. To achieve this, CMS adopted silicon detector technology. The detector has about 200 m^2 of active silicon area which makes the CMS tracker the largest silicon tracker in the world.

The tracker layout is as follow and shows in **Figure 2.14**;

- Three hybrid pixel detector modules (at radii of 4.4, 7.3 and 10.2 cm) surround the interaction point. It also has two-pixel module disks on each side. In total, the pixel detector has an area of $1m^{-2}$ and contains 66 million pixels
- The silicon-strip tracker covers the radial region between 20 cm and 116 cm and has three subsystems. The Tracker Inner Barrel (TIB) and Tracker Inner Disk (TID) extend in radius to 55 cm and are composed of 4 barrel layers with three

disks at each end. Both TIB and TIC are surrounded by the Tracker Outer Barrel (TOB)

- The Tracker EndCap (TEC), covers the region of 124 cm < $|z|$< 282 cm and 22.5 cm < $|r|$ < 113.5 cm. It compromises 9 disks, carrying up to 7 rings of silicon micro-strip.

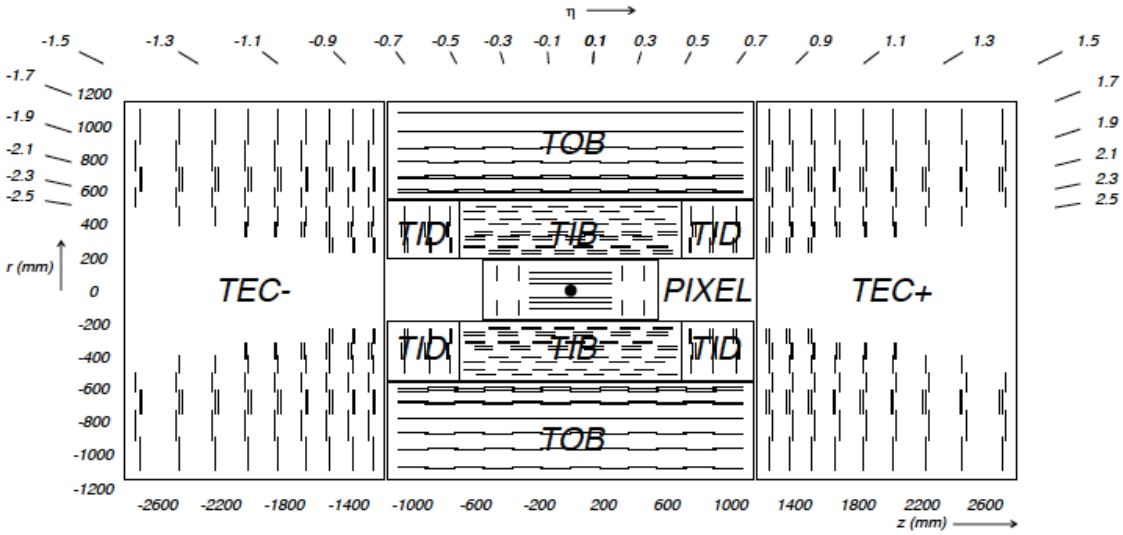


Figure 2.14: The schematic diagram of the CMS silicon inner tracker, (CMS Collaboration, 2008).

2.3.4 CMS Electromagnetic Calorimeter

Similar to the inner tracker, the electromagnetic calorimeter (ECAL) has to have fine granularity, fast response time and be resistant to high radiation damage. This can be achieved by using lead tungstate (PbW04), a homogenous high-density crystal. In the barrel section, the CMS ECAL consists of crystal with avalanche photodiodes (APD) as the photodetector. In the endcap, it consists of a preshower detector, vacuum phototriodes (VPTs) with same crystal material. One of the design criteria for the ECAL is that is can detect the diphoton decay of the Higgs boson.

2.3.5 CMS Hadron Calorimeter

The CMS Hadron Calorimeter (HCAL) consists of four subsystems, which are: the HCAl Barrel (HB), HCAL Endcap (HE), HCAL Outer (HO) and HCAL Forward (HF), shown in **Figure 2.15**. Both HB and HE are located inside the CMS solenoid, immersed in 4T of magnetic field during operation. Both of these HCAL systems are sampling calorimeters (7% sampling) with the brass as the absorber material and scintillator as the active material. To catch energy deposited beyond the HB, the HO is placed surrounding the solenoid and consisted of several layers of scintillator. The HF covers the forward region and it is made of quartz fibre as the active medium and steel as the absorber, so that it can with stand the harsh radiation damage, (CMS Collaboration, 2008, 2010b)

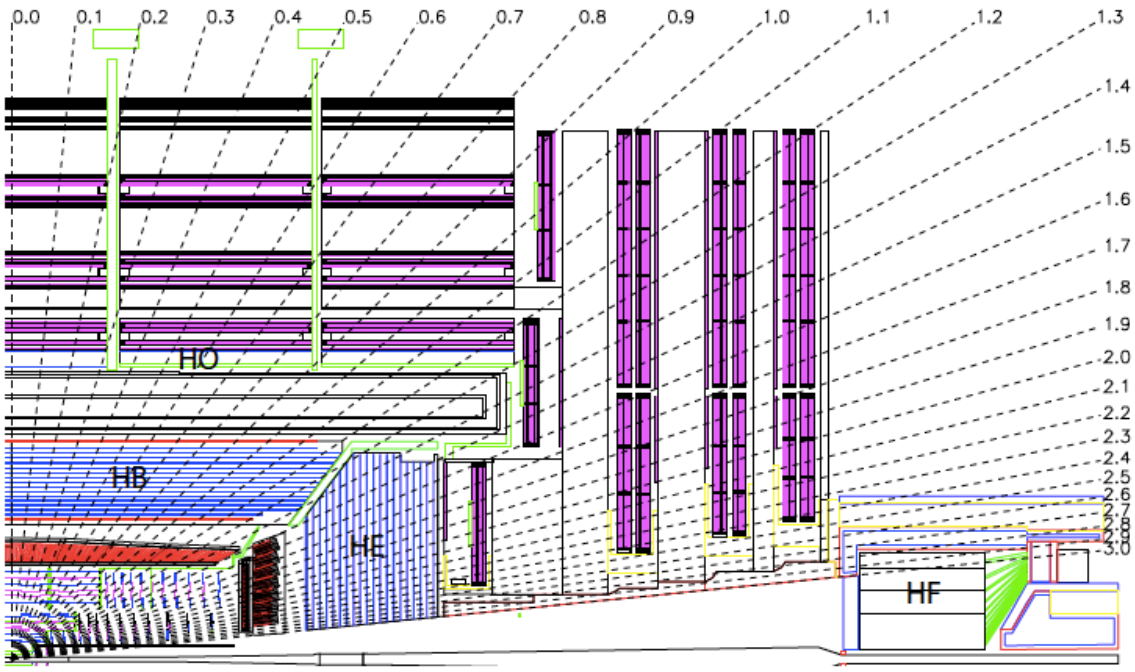


Figure 2.15: A schematic of the CMS HCAL and its four subsystems.

2.3.6 CMS Muon System

According to the CMS technical design report (CMS Collaboration, 2016), muons are the signature of most physics that is investigated at the LHC experiments. This requires the CMS detector to be able to identify (trigger) and reconstruct the muon while operating at high luminosity. In addition to that, most events of interest will have multiple muons at higher rapidity, thus, the CMS muon system must be also able to detect and measure muon over a broad range of angle. The role of the CMS muon system can be summarized into three task: muon identification, muon measurement and triggering, (CMS Collaboration, 2008)

The CMS muon system uses three different detectors, which are; the drift tubes (DT) in barrel region, the cathode strips chambers (CSC) in the endcap region and the resistive plate chambers (RPC) in both the barrel and endcap, Figure 2.15. All the muon chambers are orientated to be perpendicular to the muon trajectories and provide a hermetic coverage for the η range from 0.0 to 2.4, **Figure 2.16** & **Figure 2.17**. The summary regarding the three muon detectors is given in **Table 2.6**.

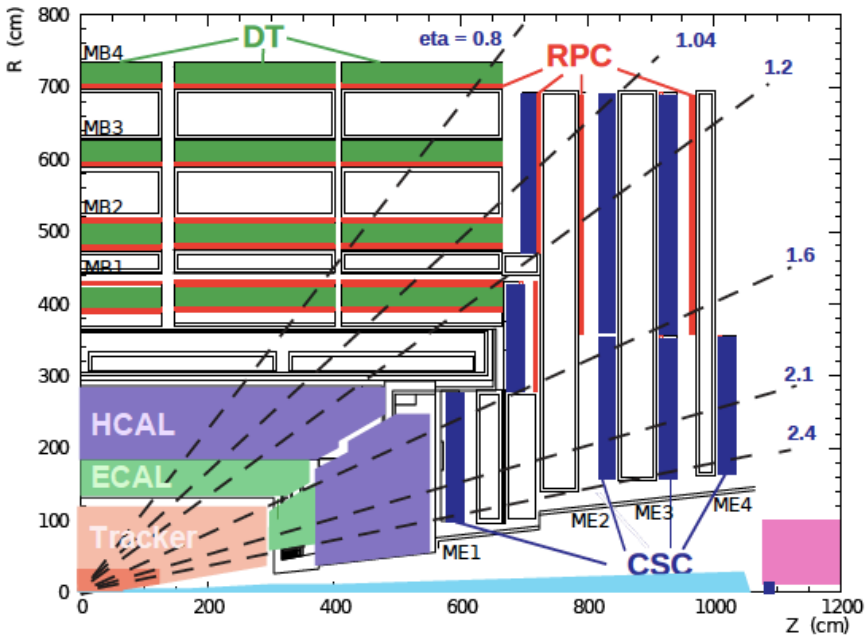
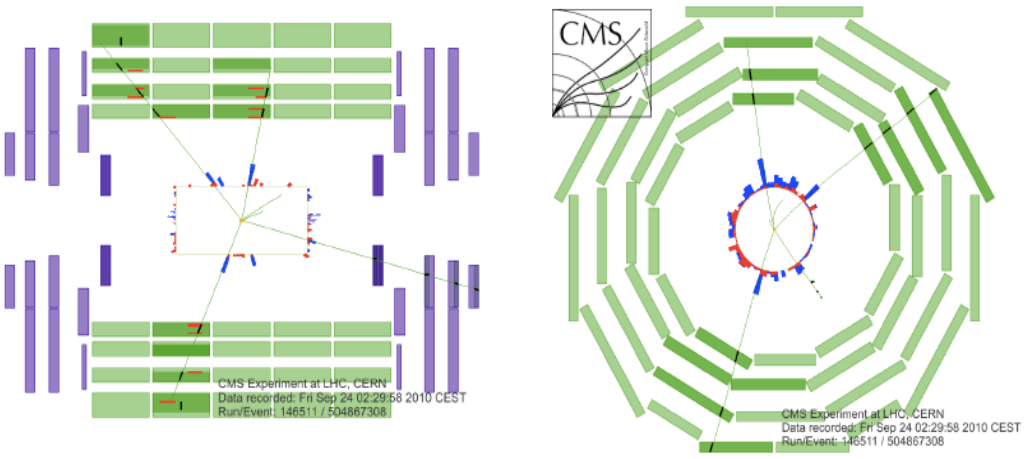


Figure 2.16: A schematic diagram of a quadrant lay out for the CMS detector showing the position of drift tube (DT), resistive plate chamber (RPC) and cathode strip chambers (CSC). Taken from (CMS Collaboration, 2012b)



(a) (b)

Figure 2.17: A longitudinal view (a) and a transverse view (b) of an event with 4 reconstructed muon tracks, using all three muon systems. All the muon detectors are orientated perpendicular to the muon trajectories. Taken from (CMS Collaboration, 2012b)

Table 2.6: A summary regarding the three muon chambers of the CMS detector, taken from CMS Collaboration,(2016)

Detector	DT	CSC	RPC
Function	Tracking pT Trigger BXID	Tracking pT Trigger BXID	BXID pT Trigger Resolve tracking ambiguities
η Region	0.0 - 1.3	0.9 - 2.4	0.0 - 2.1
Station number	4	4	Barrel – 6, Endcap - 4
Chamber	250	540	Barrel – 360, Endcap - 252
Time Resolution	5 ns	6 ns	3 ns

2.3.7 CMS TriDAS

As stated before, the LHC will produce a luminosity up to 10^{34}cm-2s-1, leading to an average of 1000 particle from 20 overlapping proton-proton collisions every 25 ns. It will be impossible to permanently store all the recorded responses coming from all the CMS subdetector as this rate. Thus, the CMS Trigger and Data Acquisition System (TriDAS) drastically reduces the recorded bandwidth to a manageable amount. According to the CMS Collaboration, (2008), The CMS TriDAS system is the combination of the Level-1 (L1) trigger, High-Level Trigger (HLT) system and the Data Acquisition System. The L1 trigger is largely custom-designed programmable electronic hardware which gives a maximal output rate of 30kHz.

The HLT are software systems implemented on a high-performance computing cluster with about 1000 commercial processers. Since the HLT takes the read out from the L1 trigger, it can perform complex calculations similar to offline analysis. The combination of L1-trigger and HLT is designed to reduce the data by at least a factor of

106 (CMS Collaboration, 2008). The Data Acquisition System manages the flow of information between the L1-trigger to the HLT.

2.3.8 CMS Detector Conclusion

The CMS detector is one the four large detectors at the LHC accelerator. It consists of an all-silicon pixel-strip tracker, a lead-tungstate scintillating-crystal electro-calorimeter, a brass-scintillator sampling hadron calorimeter and several muon detectors. A large-bore superconducting solenoid provides a homogenous magnetic field at a magnitude of 4 Tesla while an iron yoke returns the field. All parts of the detector are designed to be able to operate in a high radiation environment and provide high-efficiency responses throughout the lifetime of the detector.

CHAPTER 3
COMPUTATIONAL DEVELOPMENT FOR CMS AND SIFIR

3.1 Chapter Introduction

During run 1, the various subdetectors in CMS had to record events coming from an average of 1000 particles from 20 overlapping proton-proton bunches every 25 ns. Thus, these subdetectors generated a massive amount of data. The recorded data were then filtered and reduced by the CMS High-Level Trigger farm before being permanently stored for further analysis. In order to store and analyse these data, the CMS collaboration adopted the grid-computing technology, which combined the computing resources in several sites into one common shared resource.

This chapter focuses on the CMS high-performance computing environment with the content divided into three sections; starting with a brief review of the CMS grid-computing architecture, follow by a discussion pertaining to the CMS Remote Analysis Builder (CRAB), and the author's contribution to the development of this software. The last section of the chapter describes a High-Performance Computing (HPC) Cluster developed in Universiti Malaya called 'sifir' and the author's contribution in developing this cluster. The sifir computing cluster had been vital in this research as it was the primary hardware used to execute machine learning algorithms developed for this research.

3.2 CMS Computing Challenges

Based on CMS Collaboration (2008), the computing system architecture was designed to tackle four challenges;

- *Large scale data*: The physics done at CMS required a large statistics dataset in finding rare signals, thus it required enormous volumes of data. This required a system that was efficient in data reduction and pattern recognition.
- *High flexibility*: The system must be able to let a user access any data item during the lifetime of the experiment, as well as to support a variety of analysis methods that evolve along the goals of the experiment.
- *Manageability*: The complex and extensive computing system must be designed in a way that allows the software and the hardware to be maintained and monitored.
- *Longevity*: The system must be sustainable for 15 years or more; adaptive to changes in hardware, software, and personnel.

3.2.1 CMS Grid-Computing Infrastructure

In order to provide a large-scale computing resource for storing and analysing all the data produced by the CMS detector, the collaboration adopted the grid computing infrastructure. The CMS grid computing infrastructure is built on top of the Worldwide LHC Computing Grid (WLCG) infrastructure (Bird et al., 2005) and uses the middleware provided by the European Grid Initiative (EGI), Advanced Resource Connector (ARC) by NorduGrid (Ellert et al., 2007), and Opens Science Grid (OSG) (Pordes et al., 2007).

The CMS computing sites are classified based on their tier level; from Tier-0 to Tier-3. According to Bloom (2015), CMS has approximately seven Tier-1 sites and 50 Tier-2 sites. The role played by each tier is given in the following:

Tier-0: The CERN computing site itself, which records raw data directly from the detector and archives them as a cold backup on tape.

Tier-1: Copies the raw data from Tier-0 as custodial replica. It provides a central processing unit (CPU) farm, as well as a mass storage system (MSS) on both disk and tape with 24/7 support operation. Since Tier-1 sites have both the data and the computing resource, this site is capable of data production from computer simulation, as well as experimental data reconstruction. Only in Run-2, analysis jobs are also executed in Tier-1 sites (K. Bloom, 2015).

Tier-2: Different to Tier-1 sites, sites at this level are not required to provide tape storage services and only provide computing support during business hours. Computing-intensive jobs with low input/output (I/O) bound run more efficiently here, so that simulation production and analysis jobs are executed here. Only in Run-2, were simulation reconstructions are also executed in Tier-2 sites (K. Bloom, 2015). The University Malaya's sifir is in Tier-2 site.

Tier-3: Volunteer site that is more focused on user analysis (Mascheroni et al., 2015). (A volunteering site are computing site that are not bound with the CMS to give specific resource at all time)

CMS also has a high-performance computing cluster called the High-level Trigger farm that is used to filter irrelevant events during data taking and can also be used as an *opportunistic computing* resource during a technical stop. (An opportunistic computing resource is a resource that is only available in a certain period. For example, if all computers in a computer-lab are idle at night, this computer can be used as an opportunistic resource to provide additional computing resource for some other means. See Thain, 2005, for discussion of opportunistic computing)

In term of computing network, all sites except Tier-0 sites, are interconnected in a full-mesh network topology through a general purpose scientific network. Meanwhile, Tier-0 is connected to Tier-1 sites by a dedicated network called the LHC Optical Private Network (LHCOPN), as illustrated in **Figure 3.1**.

3.2.2 CMS Computing Activity During Run 1

According to Adelman et al., (2014), during LHC Run 1 alone, the CMS collaboration through its computing division recorded around 10 Billion data and 15 billion simulated events, while Boudoul et al., (2015) reported only 12 billon simulated events, which required the management of 100,000 processor cores and 100 petabytes of storage. Additionally, between 2010 and 2014, an average of 32.76 petabyte of data were transferred between the CMS computing sites annually (Wildish, 2015) and up to 500 active concurrent users were handled at any given time (Adelman et al., 2014). Besides, an average of 18,500 and 54,300 computing jobs were run on Tier-1 and Tier-2 sites respectively during Run 1 (K. Bloom, 2015).

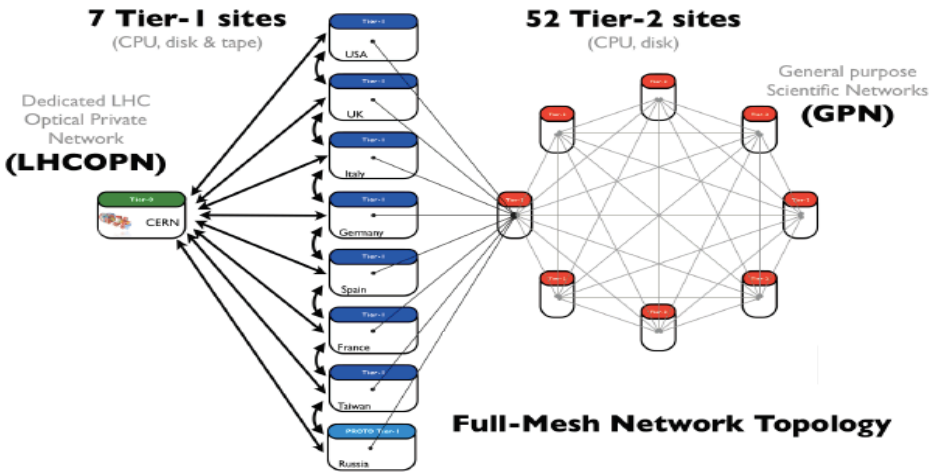


Figure 3.1: The network topology of CMS computing sites according to Tier. All Tier-1 and Tier-2 sites are interconnected with each other.

3.2.3 CMS Grid-Computing Software

The CMS computing division has three primary tasks; data storage, transfer, and analysis. In order to perform these tasks, a variety of software programs are adopted into the computing environment or developed in-house when they are required. **Table 3.1** lists the primary software programs that are utilized in the CMS computing based on their roles.

Only software from the analysis and submission infrastructure group are relevant for this research, hence the detail regarding the software from these two groups will be given in the subsequent subchapter. An in-depth study of each middleware is outside the scope of this thesis; however, the interested reader is directed to the article given in the references.

3.2.4 CMS Submission Infrastructure

All computing jobs submitted to the CMS grid-computing infrastructure originate from two workflows; analysis and production workflows. Production workflow is reserved for simulation and event reconstructions that are managed and produced centrally by a few experts (Adelman et al., 2014), while user analysis and private simulation jobs are submitted through the analysis workflow channel (Mascheroni et al., 2015).

The front end for the production and the analysis workflow are the Workload Management Agent (WMAgent) and the CMS Remote Analysis Builder (CRAB) respectively (Boudoul et al., 2015; Mascheroni et al., 2015). Nevertheless, these two workflows use the same computing resource, thus, they are merged at the submission infrastructure level, as shown in **Figure 3.2**. According to Belforte et al., (2014), in the initial phase of the CMS experiment, direct submission to the computing node (in various sites) or an intermediate workload management system is used. The dominant high-level scheduler in Europe (Marco et al., 2009) is the gLite WMS while the HTCondor G scheduler is dominant in the United States of America (Thain, Tannenbaum, & Livny, 2005).

However, the direct submission method has been proven to be inefficient during Run 1 due to its high dependence on the network and the instability of a site's middleware. Furthermore, in Run 2, different computing architectures, such as cloud, local batch, and opportunistic resources, are incorporated into the overall CMS computing resources (Balcas et al., 2015). Furthermore, the single submission entry point eliminates the need for complex job prioritization at the computing node, since it could be collectively

controlled at the central management level. Hence, there is a strong need for a single submission infrastructure that can control all (global) computing resources.

Therefore, CMS has moved towards a unitary submission method, which is call the glideinWMS global pool (Belforte et al., 2014), as depicted in **Figure 3.2**. The system is based on 'pilot', in which a lightweight job is first submitted to a compute element at a site. Only when the pilot job starts to execute in the compute element will the actually analysis job be run. The glideinWMS is based on the HTCondor system, and according to Fajardo et al., (2015), it can manage up to 200,000 simultaneously running jobs for a single internationally distributed dynamic pool.

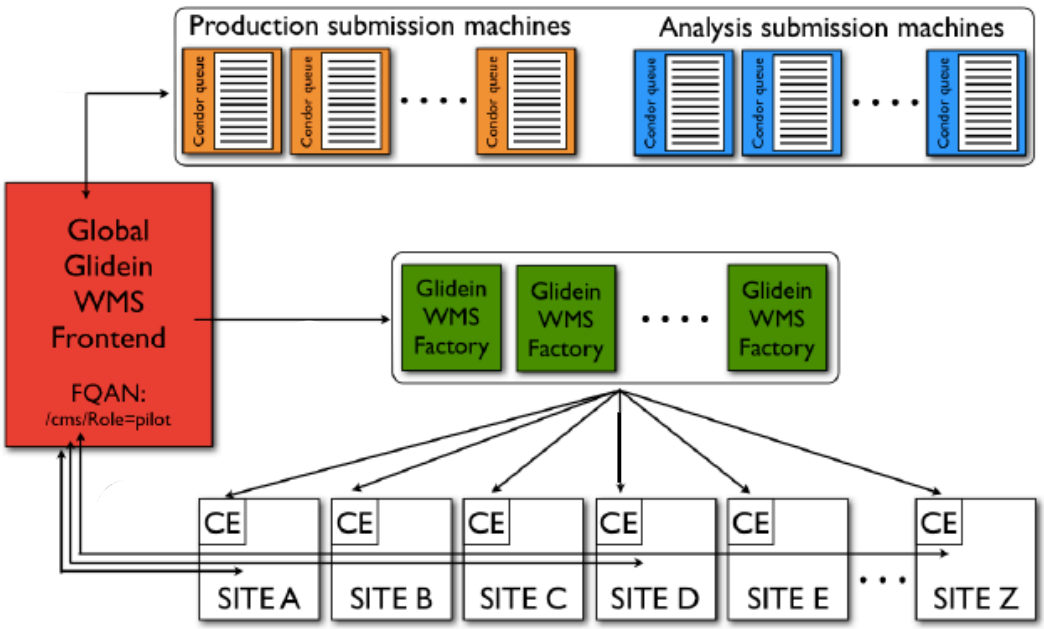


Figure 3.2: The general system design for the CMS glideinWMS global pool (taken from Belforte et al., 2014)

Purpose	Function	Software	Reference
Transfer	Transfer data between sites	Physic Experiment Data Export (PhEDEx), Data Aggregation System (DAS), and Any Data Any Time Anywhere (AAA)	Giffels et al.,(2014), Bloom et al., (2015)
Bookkeeping	Files and dataset metadata catalogued	CMS Dataset Bookkeeping System (DBS)	Giffels et al.,(2014)
Constant	Information on alignment and calibration constant	Frontier, SQUID	Blumenfeld et al., (2008)
Software Distribution	Allows software to be accessed on all sites	CERN Virtual Machine Software Application (CVFMS)	Buncic et al., (2010)
Submission	Submits computing job to sites	gLite, HTCondor_G, glidein	Belforte et al., (2014), Marco et al., (2009), Sfiligoi et al., (2009), Thain et al., (2005)
Stress Test	Does stress test on sites	Hammercloud	Ster et al., (2011)
Monitoring	Monitors site, job, and transfer	Dashboard, WLCG SAM, SiteDB	Andreeva et al., (2008), Metson et al., (2010)
Production System	Centrally produces simulation workflow	Workload Management Agent (WMAgent)	Fajardo et al., (2012)
Analysis System	User-orientated job submission	CMS Remote Analysis Builder (CRAB), AsyncStageOut(Run2)	Mascheroni et al., (2015), Riahi et al., (2015)

Table 3.1: The various middleware applications used in the CMS computing environment in Run1 and Run 2 sources: Adelman et al., (2014)

3.3 Introduction to CMS Remote Analysis Builder (CRAB)

The front end of the analysis workflow is the CMS Remote Analysis Builder (CRAB), which takes the user algorithm, configuration parameter, as well as the defined dataset, and then, manages the job execution on a remote computing element. CRAB is designed to accept a variety of analysis algorithms and input sources (hence, adaptable to changes in analysis goals), as well as in shielding the user from the technicality difficulty of CMS submission infrastructure (Mascheroni et al., 2015).

The CRAB has undergone three reincarnations with its first version in production since the Spring of 2004. Its first version was a stand-alone Python application that could be run on a user's workstations. The application handles all interactions with other CMS specific and Grid Middleware services. In CRAB2, the application adopted the server-client framework; delegating and synchronizing all operations with the CRAB2-server. However, the CRAB2-client still carried out multiple tasks, including data discovery and location via DBS and PhEDEx. It also retrieved information on how to interact with different site computing elements and storage elements via SiteDB (Cinquilli et al., 2012).

One of the key motivations of redeveloping the CRAB system from scratch to the current version of CRAB3 has been due to the high failure rate during the stage-out process. In CRAB2, once an analysis job has finished executing a remote computing element (CE), it will try to stage-out the job output directly from the CE to a user-defined storage element. In a grid computing environment, a user storage element can be physically located across the globe from the CE. Thus, the stage-out process can take valuable computational time, rendering CE in an idle state until the process is complete.

Furthermore, the direct stage-out process is prone to failure due to grid middleware and network instability. If the stage-out process fails, the whole analysis job has to be repeated (Adelman et al., 2014; Mascheroni et al., 2015). From here, the discussion focuses only on CRAB3 implementation and the term CRAB refers to the CRAB3 version.

3.3.1 User Perspective

CRAB is a command-line-based application that runs on the CERN ssh-terminal node. Once a user has logged onto the node, the user only needs to deploy the CMS Software (CMSSW) environment via the command line `cmsenv` and export the CRAB source-code path available in the CMS CVFMS. Having done that, the user now can use multiple `crab` commands as shown in **Table 3.2**;

Table 3.2: List of `crab` commands available for user as of January 2016

Command	Action
`submit`	The application takes a user-defined configuration file and creates a task on the grid base on the parameter in the configuration
`status`	Retrieves information of all or a given submitted task
`kill`	Kills a given task that has been submitted
`getlog`	Retrieves the log file of a given task
`getoutput`	Retrieves the output of a given task
`resubmit`	Resubmits a failed task
`proceed`	Continues submission that is previously in a 'dryrun' mode
`purge`	Clears a user's sandbox (a cache area for user's code)
`remake`	Recreates a user's task information in the local directory
`report`	Retrieves the list of good 'lumi' section of a given task
`checkusername`	Validates if the user's username is identical with the information in SiteDB
`checkwrite`	Validates if the user has read/written access to a given site storage element
`uploadlog`	Uploads a crab log file to the developer in case of bug

3.3.2 CRAB Architecture

Cinquilli et al., (2012) designed the initial CRAB3 architecture and a majority of the initial design still holds in the current version (as of January 2016), including:

1. A central service without any single point failure
2. Asynchronous stage-out implementation
3. A HTTP transaction at the interface (RESTful Interface)
4. A central global queue
5. A distributed agent processing the workflow

6. Global monitoring

Points 4, 5, and 6 are implemented through the glideinWMS global pool, as explained in subchapter 3.2.4, whereas points 1, 2, and 3 are implemented on the CRAB3 system. The CRAB3 system is divided into multiple components, which are:

CRABClient: The interface between the user and the rest of the CRAB system, where it receives all the crab commands listed in **Table 3.2** from the user. It is written fully in the Python language and it is designed to be lightweight, parsing, and receiving information from the CRABserver or other CMS service through the Python-pycurl module in JavaScript Object Notation (JSON) form. The client also validates the parameters given by the users before submitting them to other services.

CRABServer: Acts as the front end (gateway) for a user's request and handles user authentication. The request is validated again before submitting it as an entry into the task database; the current implementation uses, and Oracle database, nevertheless, the component is also compatible with the MySQL database.

TaskWorker: It checks the CRABServer database at a regular period of time for a new request; it executes the command given by user via CRABClient such as submission, kills, and resubmits. The TaskWorker has the capability to spawn multiple slave-processes in parallel, with each slave-process responsible for executing only one unit of work on a task. For example, the job submission command requires the slave process to initiate data discovery by using DBS, job splitting, and then send the full description of the task to the HTConder scheduler. The component also stores the status of each task and synchronizes it with another task worker instance so that the same task is not executed twice.

AsynchronousStageOut *(ASO)*: The stage out process of a user's job output happens in two steps: 1) First, the output is copied to a local temporary storage area in the execution site. 2) An ASO component is notified to do the transfer via File Transfer Protocol (FTP) on a predefined channel, similar to PhEDEx. Once the transfer is over, the ASO would update the DBS to specify the final location of the output. A full description of the ASO is given in Riahi et al., (2015).

Each job submission from CRAB to the CE is wrapped in a job wrapper script. The script:

1. Sets up the CMSSW environment at the CE
2. Executes the user's code
3. Records monitoring information
4. Reports any error during execution
5. Initiates the output transfer process.

The CRAB components, along with the scheduler, and CE are shown in **Figure 3.3**.

Each component can have multiple instances running simultaneously to create load balancing between instance and redundancy in the overall system. Each component can also be developed independently from each other as long as the interface between them is consistent. Since the component is also centrally managed, rolling out an updated version (with added features and to eliminate bugs) is done automatically without the user's participation.

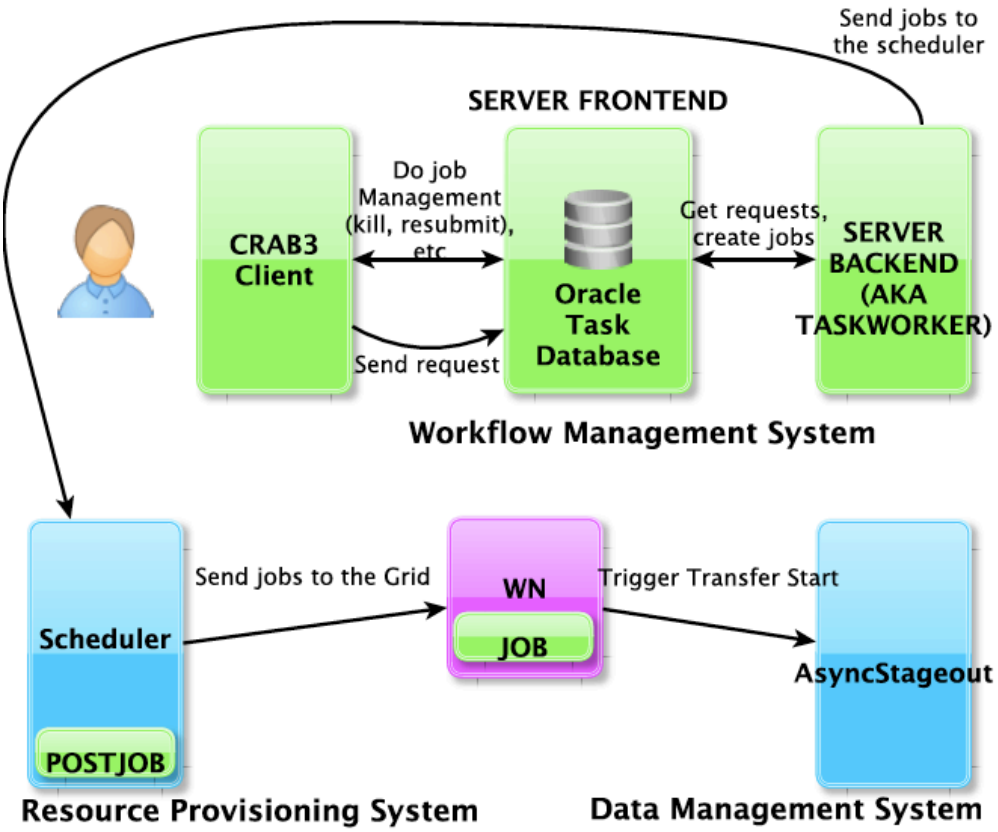


Figure 3.3: The CRAB3 Components and the internal mechanism of the job submission (taken from Mascheroni et al., 2015)

3.3.3 User Adoption

Figure 3.4 shows that the number of unique users steadily increased from Jun 2014 (declared production ready) until May 2015. The last release of CRAB2 was issued in November 2014, and after that, CRAB3 has become the primary front end for CMS users to submit their jobs to the grid computing infrastructure.

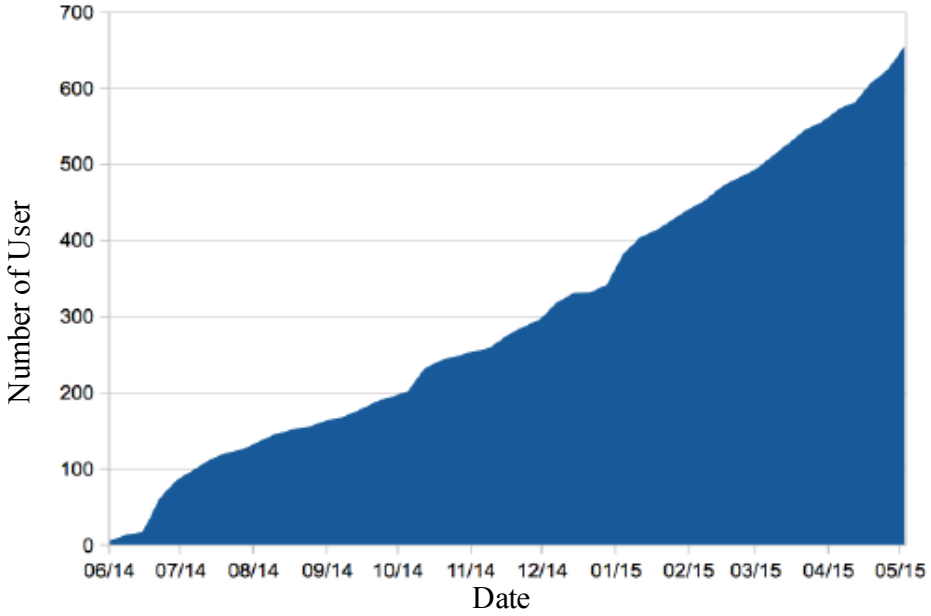


Figure 3.4: The cumulative number of users using CRAB3 from Jun 2014 until May 2015 (taken from Mascheroni et al., 2015)

3.4 Development of CRAB3

The author's contribution to the development of CRAB3 was carried out between October 2013 and September 2014. A total of 66 improvements (Github pull request) had been merged into the main CRAB3 production algorithm. Considering that the contribution period was carried out during the LHC's first Long Shutdown and the CRAB3 was still in the pre-production phase, most of the development done was focused on creating a stable release, as well as adding new features present in CRAB2, but not yet in the CRAB3. The subsequent subchapter provides the details pertaining to the primary development contribution to the CRAB3 system.

Contributed codes can be accessed at: https://github.com/qunox/CRABClient and https://github.com/qunox/CRABServer.

3.4.1 CRAB3 Error Report Mechanism

As stated before, one of the main goals during this development period was to create a stable release of CRAB3, and at the same time, to include new features. These two goals are working against each other as creating new features tends to introduce an unexpected bug into the code that is meant to be stable. Thus, one of the main contributions to CRAB3 was to create a rapid semi-automatic feedback mechanism for handling unexpected errors.

If the CRABClient component encounters an unexpected error, an internal mechanism is flagged, which automatically uploads the user's CRABClient log file to the CRABServer. A CRAB3 developer will then receive the notification regarding the error and fetch the user tarball and log file for debugging. In the event that a user complains that a bug has occurred, but the CRABClient fails to notice it, the user can still notify the developer and upload the CRABClient log file manually by using the `crab uploadlog` command.

This feedback mechanism had been a major development effort due to the following reasons;

1. In the original logging mechanism, nearly half of all the unexpected errors occurs before the logging module is initiated, thus, no recording of the error is done and debugging in this situation is impossible. Even if the error is logged, the full information about the error were not given. Hence, the internal logging mechanism of the CRABClient component have to be re-written from scratch.
2. Reproducing the error experienced by the user requires the developer to have enough information, such as the user's job configuration, terminal environment

parameters, as well as the CMSSW version. This information was not available initially, hence, a mechanism in which all the necessary information was uploaded when the user submits a job was also developed.

3. A new crab command: `crab uploadlog` was created so that a user can manually upload the log file.

The code structure for the automatic auto-upload-log is given in **Table 3.3**;

Table 3.3: The code structure for crab auto-upload-log

`crab` auto-upload-log code

if unexpected error flag == True:
  if user proxy file exists && log file exists:
  get CRABServer instance in use URL (base URL)
  get user cache address on CRABServer with the given proxy and base URL
  upload log file to user cache
  else:
  Advice user to upload log manually

3.4.2 Validation of User Read/Write Access

One of the key policies in the CMS collaboration is that each user must have his/her personal storage area maintained by the home institution. Thus, each user can only permanently stage-out their analysis output to his/her particular storage area. Furthermore, the storage address (pathway) also must be discoverable by the PhEDEx

and the DBS service for CRAB to create the necessary Logical Filename (LFN) and Physical Filename (PFN) for the stage-out process.

Moreover, it is common for a user to be unaware of their read/write access to a particular storage and/or their LFN address differs from that registered under PhEDEx and DBS. Both these situations raise an exception (error) during the stage-out process by CRAB.

The `crab checkwrite` command allows a user to validate if they have read/write access to a particular storage area. This was a major development because:

1. Failed stage-out creates a very inefficient usage of the grid computing resources since the analysis job that has been computed is useless. This error was common during the CRAB3 pre-production phase.
2. The `crab checkwrite` command establishes a direct I/0 bond between the CRAB3 service and the user storage element. Establishing such connection requires proxy delegation mechanism for authentication purpose, which is a complex process. On top of that, the command also requires proxy for the LFN validation by PhEDEx services. Thus, the consistency of API between various CMS computing services has to be maintained.
3. During the First Long Shutdown, there was no standardization between CMS computing sites on which read/write protocol to be use. Different sites used different versions of the Storage Resource Manager (SRM). The management of contain in the grid-computing storage are done by using one of two Linux command which is the `lcg` command family (`lcg-ls`, `lcg-cp`, and `lcg-del`, (Gross, 2012), and the `gfal` command family (`gfal-copy`, `gfal-ls`, and `gfal-rm`, ("GFAL2 utility tools," 2016). The `gfal` command was used to

replace the `lcg` due to the `lcg` command have consistency error between its version.

The mechanism for the `crab checkwrite` in its original version is given in **Table 3.4**;

Table 3.4: The code structure for the `crab checkwrite`

`crab checkwrite` code

Retrieves user CERN proxy ('myproxy') based on the user's role and group
Creates a dummy file in the user local space
Retrieves the user's CERN username from the DBS service
Retrieves user storage PFN from the PhEDEx service
Tries to copy the dummy file to the given PFN by using lgcp -srmv2 command
if exitcode == 0:
Tries to delete the dummy file by using lg-del -srmv2
else if exitcode != 0 && exiterror == timeout error:
prints 'Try again latter'
else if exitcode != 0 && exiterror == dummy file already exist error:
Tries to delete the dummy file by using lg-del -srmv2
Tries to copy the dummy file again by using lgcp -srmv2 command
exitcode = new exitcode
if exitcode == 0:
return User is able to write on the given PFN by using the given proxy
else:
return User is unable to write on the given PFN by using the given proxy

3.4.3 Parallel Remote Copy

The `crab getoutput` and `crab getlog` command use the same module, call remotecopy.py to copy an output file from the sites where the analysis job was executed. The user commonly uses these two crab commands for debugging their analysis algorithm

In the original version of remotecopy.py, files are copied to the user local space in a serial manner, resulting in a very slow (almost crawling) download speed. Hence, a change to parallel downloading was deemed necessary.

This development was initially considered to be minor, but progressively grew to be a substantial development effort. The root of the complexity was caused by the need for the module to be re-written, while retaining certain parts of the algorithm due to legacy issues.

Complexity also arose during the initial test of the parallel download mechanism, as it was discovered that if the number of parallel connections per user was set too high, it resulted in multiple connection attempts by multiple users (all originated from CERN) to one single execution site; such massive-multiple connection attempts were deemed to be a Denial of Service Attack (DOS). In this situation, the connecting site would tend to kill all connections (connection-reset-by-peer error). Setting the number of parallel connections per user at a lower number would however contribute to poor performance since certain computing sites set low downloading speed limits per connection. In such a situation, the number of allowed connections should be high.

After several trials were made, it was determined that the default number of connections per user should be set at 10 with a maximum of 20 connections. The remote copy algorithm is given in **Table 3.5;**

Table 3.5: The code structure for the remotecopy module

remotecopy algorithm

Retrieves files PFN
Verifies if the number of connection(s) requested is appropriate
Calculates timeout period based on file size
Starts sub-process according to the number of connection(s) requested
Creates python queue manager (python inter-process piping manager)
In parallel:
Sub-process takes a file PFN
Tries to download file with the given PFN
if the file is downloaded:
Do checksum on the download file
if checksum fails, try downloading again
else:
Flag as failed to download
Report successful download or failed download.

3.4.4 CRABClient API

During the pre-production of CRAB3, the 'multiCRAB' had been one of the most requested features to be ported from CRAB2 into CRAB3. The idea behind multiCRAB is to give the user the ability to submit and to monitor multiple analysis jobs in one single command line. In CRAB3, the idea of multiCRAB is expanded to become a Python

Application Programming Interface (API) that allows a user to develop their own job management script. Thus, the CRABClient API development project was initiated.

By employing the CRABClient API, users are expected to write a python script for a job submission, such as below;

```
from __future__ import division
from CRAB3 import Command, ClientLogger

log = ClientLogger.ClientLogger().add()
cmd = Command.Command()

crabconfigpath = '~/foo/cranbconfig.py'
cmd('submit',crabconfigpath)
```

In this design, the `Command()` function input is exactly similar to the CRABClient application command (as shown in **Table 3.1**), as well as the command line parameter. Thus, users need not memorize a new set of commands when using the CRABClient API.

Examples of case scenarios for the CRABClient API are;

- *Serial analysis*: An analysis routine that takes the product of the previous analysis as the input of the next analysis job; by using the CRABClient API, a script can be made to create atomization of task management.
- *Optimization analysis*: User submits a batch of analysis job with each job having a different value for a certain parameter. Batch submission and monitoring can be easily developed by the user via CRABClient API.
- *Corrupt sector analysis*: In some instance, a file may contain certain recording error that could not be read by the computer (corrupted sector). A batch of

submissions is executed on a small sector from a large input source. The sector becomes smaller on each batch submission until the exact corrupted sector is pinpointed. The CRABClient API allows users to automatize such task.

The CRABClient API consists of only two additional modules on top of the original CRABClient application library. They are the logger (ClientLogger.py) and the command module (Command.py). The logger gives the user the ability to customize various logging criteria, such as the logging level and the output path. It is worth noting error logging was crucial for debugging purposes.

The command module acts as an interface between the CRABClient API and the CRABClient application library. It reinterprets the command given by the API side into a set of inputs that is understandable by a module in the CRABClient library. It also handles and translates errors and exceptions raised by the CRAB or other CMS services to a message that can be understood by the user. Minor development effort was also done on the CRABClient library to accept the input from the CRABClient API.

3.4.5 Minor Development Contribution

Other minor development contributions to CRAB3 include;

`crab purge`: Each CRAB3 user has a cache space on the CRAB3 Server that is used to store the user's submitted job tarball. This cache tends to fill up quickly for super users who regularly submit jobs or have analysis code that is large. A user cannot submit any additional job if his/her cache area is already full. The command contacts the CRAB3 Server to flush the user's cache system so that the user can submit additional job(s).

`crab remake`: Certain crab commands, such as `crab status` and `crab kill`, take a file in the user local directory that contains the necessary information about a given task. This file is lightweight and it is created during task submission. Nevertheless, in case the user accidently deletes this file, the `crab remake` draws available information about the given task from the CRABServer and remakes the file.

Multiple shells: In the original version of CRAB, a run-time bug will occur if a user runs CRAB3 in multiple shells. This happens because the multiple CRAB instance tries to read/write on the same temporary-output file, causing inconsistency. The resolution is simply to have multiple copies of this temporary-output file for every shell the user opens.

Nonsynchronous submission: This parse command offers the ability for a user to submit and wait until his/her submission is marked as successful. This eliminates the need for the user to keep giving `crab status` command to the CRAB Server to check if his/her submission has been accepted.

Force ASCII character: This improvement is minor, but proved to be important. CMS Collaboration members come from across the globe and use various types of keyboards, where some keyboards have Non-ASCII characters on them. Therefore, a user who gives Non-ASCII characters in the CRAB configuration file tends to raise an unexpected error in CRAB3. Thus, a filter was introduced in CRAB to refuse any input with Non-ASCII characters.

3.4.6 CRAB3 Development Conclusion

The CMS Remote Analysis Builder (CRAB) is the front end of the analysis workflow; it provides a way for physicists to submit their analyses or private simulation jobs to the CMS grid-computing infrastructure. In its 3rd version, the CRAB3 adopts the client-server model and uses the Asynchronous Stageout (ASO) method to minimize stageout error that was common in CRAB2. The author has contributed several improvements to the system, including: rewriting the CRAB3 error reporting mechanism, a method for users to validate their read/write access (`crab checkwrite`); introducing parallel remotecopy; and creating the CRABClient API. Meanwhile, the minor improvements made to the CRAB3 system include; `crab purge`, `crab remake`, multiple shell instances, nonsynchronous submission, and force ASCII character.

As stated before, the primary objective of this research had been to investigate the application of a machine-learning algorithm for particle physics analysis. The original plan was to use the developed CRABClient API to encapsulate the machine-learning code so that it can run on the CMS grid-computing. This method is possible given that CRAB is designed to be flexible in accepting any privately developed analysis code. However, the CMS computing element does not support the multicore execution that is necessary for the machine-learning code. Thus, the original plan to use CRAB and CMS grid computing was abandoned and the project was shifted to run solely on the UM High-performance computing cluster, sifir.

3.5 The sifir Initiative

The CMS Collaboration expects each user to have a storage area provided by their respective home institute. Thus, it is an indirect requirement for University Malaya (UM) to have a computing site that is linked to the CMS grid computing network. Under the National Centre for Particle Physics (NCPP) and the University Malaya High-Performance Computing Centre (UMHPC), the *sifir* project was born in the middle of 2014. sifir is a computing cluster developed with the following objectives;

- To become a Tier-2 site for CMS grid computing
- To become one of the participating sites for the Academic Grid Malaysia
- To provide central computing resources for the UM research need.

Therefore, the computing resource in sifir is shared among three organizations; CMS collaboration, Academic Grid Malaya, and the UM community. In this research, sifir was employed as the main computing hardware used to execute the developed machine-learning code.

3.5.1 sifir Original System Architecture

sifir is a general-purpose high-performance computing (HPC) cluster that uses the Linux-based operating system (OS) that runs on the commodity hardware. In an addition, open source software is used to manage and monitor the cluster at all levels. The grid middleware was sourced from the European Middleware Initiative (EMI), as it is intended to be compatible with the WLCG.

sifir originally consisted of a single master node, a single general-purpose server call controller, ten worker nodes, as well as one single storage node (**Figure 3.5**). An

Ethernet router and fibre-optic switch support the cluster network backbone. Four virtual machines are placed inside the controller node, where their label and purpose are given in the following;

- ***Computing Element*** (ce.sifir.um): Houses the middleware that is related to the computing resource management, such as the Computing Resource Execution and Management (CREAM), as well as the Sun Grid Engine (SGE).
- ***Storage Element*** (se.sifir.um): Encapsulates the Disk Pool Manager (DPM) middleware that functions as a disk storage management for grid purposes.
- ***Information system*** (is.sifir.um): The Berkeley Database Information Index runs in this VM, functions as the information provider about the sifir system to the grid.
- ***User Interface*** (ui.sifir.um): The cluster login node for local users.

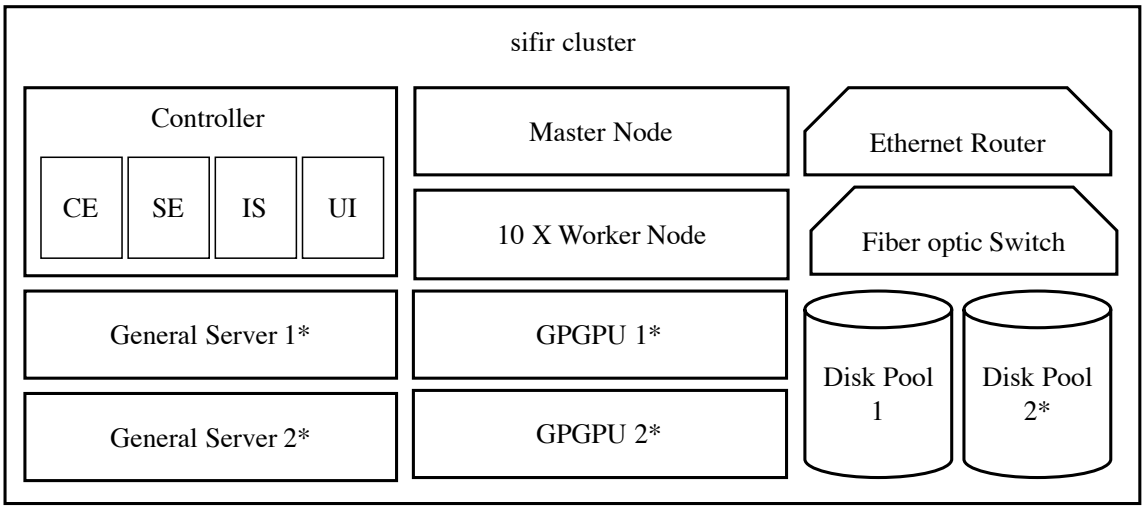


Figure 3.5: sifir cluster hardware and middleware architecture. Components marked * are additional new hardware.

However, the hardware of sifir has quickly grown to additionally have;

- Two general purpose workstations (server)

- Two machines with general-purpose computing with graphic processing unit (GPGPU)
- An additional storage node

The original system architecture with the additional hardware is shown in **Figure 3.5**;

3.5.2 Weaknesses in the Original Architecture

There are multiple weaknesses in the original design implementation, which are:

a. The main flaw of sifir in its initial period was in the network, both for the Local Area Network (LAN) and the Wide Area Network (WAN). Starting with LAN, difficulty occurs when the additional hardware (marked with * in **Figure 3.5**) did not have a fibre-optic network interface and was only connectable via Ethernet, which is 10 times slower than the fibre-optic connection. Using the Ethernet to establish I/0 connection between the new hardware and the disk pool led to degradation in the whole system network.

b. Another weakness was on the WAN network between CERN and sifir, since one of the main purposes of the cluster is to become one of the CMS Tier 2 sites. In its initial phase, the average download speed (through the `scp` command) between a CERN node and se.sifir.um node was only 33.3 kB/s. On top of that, the average `yum` (a Linux command line) update download speed for all nodes was at 18 kB/s. It was later found that the University firewall was dropping packages coming from sifir at a rate of more than 90% (**Figure 3.6**), which caused significant network instability. In terms

of the network route, the connection began from the UM internal network route to Jaring (a domestic internet service provider), then to the Malaysian Research Network (MYREN), proceeded to China's Trans-Eurasia Information Network (TIEN3), continued to the French GÉANT network, and finally reaching the CERN network (**Figure 3.6**). Moreover, three networks hops with considerable latency was required before the sifir packet reached MYREN; and hence, the connection would have been already slowed domestically.

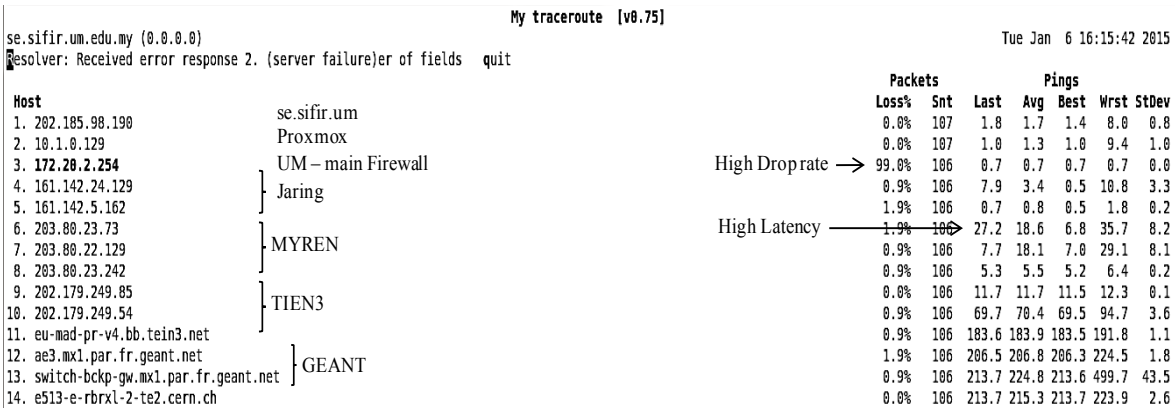


Figure 3.6: Traceroute result from the se.sifir.um to cern.ch network

c. The original sifir system architecture had been meant to be grid first, and local cluster second. Thus, the software that is needed for cluster management such as the resource provisioning software and identity management software has not been deployed properly. This create problems when local user wants to access sifir resource.

d. The login node (UI) VM had been placed on the same physical node as other critical grid services (CE, SE, and SI). However, users are also encouraged to execute their developed code on the UI before submitting it to the grid/cluster working node so that system-specific-bug can be discover before submission. This will lead to a greater

efficiency in the overall grid/cluster resources usage. On the other hand, if a high number of user test-run their codes inside the UI at the same time, a resources scarcity (including network) between UI and other services would occur. Thus, this design is not scalable as users increase. In an idle implementation, both UI and other critical services should run on a separate physical node with resource redundancy.

e. Visualization has become an integral part of computing system management. For the system admin, it can be used to create a lightweight VM as a testbed, encapsulating past middleware versions as a fallback, and fragmentizing the system to slow down intrusion due to security breach. A VM can also act as a toybox for end-user, which uses a non-standard application that is not maintained by the system admin. The original software that was used for virtualization purpose in sifir was *proxmox*. In practice, proxmox does provide stability and coherence in managing VM, however, it lacks the valuable end-user feature that is provided by other software programs, such as OpenStack and Eucalyptus. Without this end-user feature, the system admin would ultimately have to manage the end-user VM, which is not feasible if sifir were to continue scaling up its end-user numbers.

f. The original high-level scheduler for the cluster was the Sun Grid Engine (SGE), which was later made open source software through the Open Grid Scheduler; however, the SGE or Open Grid Scheduler was last updated in 2013 ("Open Grid Scheduler," 2016). It is not a best practice to use an unmaintained software since it can cause a security thread.

3.5.3 Improvement Initiative for The Sifir Cluster

The following improvements were made on the sifir cluster;

a. At the time of writing this thesis, the sifir's LAN problem had not been fully resolved since it required a fibre-optic network card to be installed on all additionally acquired nodes.

b. A Level 3 switch was installed on the sifir cluster with a physical network connection between sifir and MYREN, which considerably increased the connection speed.

c. For cluster management at the local level, the Ganglia software (Massie et al., 2012) was used for resource monitoring and the openLDAP software for identity management.

d. In order to solve problems d. and e. in subchapter 3.5.2, the OpenStack solution was adopted. In this solution, The OpenStack compute element (NOVA), identity manager (keystone), and image manager (Glance) were deployed in general server 1, while general server 2 only had Nova. The OpenStack storage component (Cinder) was planned to be deployed in the disk pool 2, in this way, VM in general servers 1 and 2 would use disk pool 2 for storage. Since OpenStack has the ability for automatic load balancing between the computing nodes, the problem of balancing resource between the UI and other critical services is solved. The OpenStack also provides end-user features through a web-based application, which provides a practical means for users to create and manage their VM, with minimum support by the system admin. In the

event of inadequate resources for additional VM deployment, scaling up the OpenStack can be easily achieved by deploying Nova on a new node.

3.5.4 Lessons Learnt in Managing an HPC Cluster for Scientific Purposes

A set of computer nodes does not become an HPC cluster due to its hardware specification, but only becomes one through great software architecture design. At the fundamental level, a computer cluster (or grid for that matter) is only several computer nodes exchanging messages between them to complete a given task. The control and the management of this inter-node message are entirely done via the designed software; thus, an HPC cluster is only good as its software implementation.

Choosing and designing an HPC system architecture is a not a small task as there is no conventional system architecture, but it does help to start the design process by knowing the limitations that are in place and the overall objective of the cluster. Limitations such as in financial resources, hardware availability, and human resources provide an indication of what can be achieved in practice. The objectives provide guidelines on what is the minimum necessary performance a cluster should provide.

Having a test bed is crucial to allow the system admin to test a solution (new feature, bug fix, patch, etc.) before deploying it into production.

3.5.5 sifir Conclusion

sifir is a high-performance computing cluster developed with several objectives, including in becoming a CMS Tier-2, as well as in becoming the primary computing cluster in University Malaya. However, the original system architecture was too grid-computing centric, and improvements were made to balance between the grid-computing need and the local-cluster computing requirement. The improvements that were made on the cluster included; upgrading the cluster WAN, as well as the deployment of OpenStack and other cluster-management middleware. With this improvement, the machine-learning code that has been developed in this research can now be executed and monitored properly on the cluster.

CHAPTER 4
MACHINE LEARNING AND THE DEVELOPMENT OF SELF-ORGANIZING MAP APPLICATIONS

Akan tetapi akal atau roh itu ialah bekas daripada perjalanan otak yang sihat laksana gejala api itu timbul daripada lilin yang sedang terbakar

Hamka, 2009, p.31

4.1 Chapter Introduction

This research uses the University Malaya High-Performance Computing Cluster, sifir, as its main hardware, as well as an internally developed application based on the Self-Organizing Map (SOM) algorithm as its main software. SOM is a type of machine learning (ML) that clusters instances without supervision.

Before going into the details of SOM, this chapter, first, introduces some ML terminologies to the reader. It, then, introduces several other ML algorithms that are relevant to this research and some instances concerning the application of these algorithms in the study of particle physics. After that, an in-depth focus regarding SOM and the development of an SOM application is given. The chapter ends with details concerning the deployment of the SOM application prototype in a cloud-computing environment.

4.1.1 Machine Learning Terminology: Supervised and Unsupervised Learning

Before discussing various ML algorithms, an introduction to common ML terminologies is given. ML algorithms are categorized by their learning methods, which include supervised learning, semi-supervised learning, unsupervised learning, and reinforced learning. In this research, only supervised and unsupervised learning is relevant and the details concerning these learning types are given in the following;

Supervised Learning: This type of learning is commonly associated with classification algorithms, in which labels are already defined before learning. The algorithm learns from a fully labelled dataset and classifies new instances based on these labels. In this process, the ML does not create a new label.

Unsupervised Learning: This type of learning is usually associated with a clustering algorithm. In clustering algorithms, no label is pre-defined before the learning process and the algorithm has to create new labels based on the patterns it discovers in the dataset. Unsupervised learning is commonly used since real-world data are usually unlabelled.

Most ML algorithms accept two kinds of inputs, hyperparameter and dataset. An explanation of hyperparameters is given in the next subchapter, whereas an explanation about datasets is provided in the following;

4.1.2 Instance and Datasets

ML requires certain inputs to learn, which are known as datasets. A ***dataset*** is a collection of data that has a defined structure like column and row. Supervised ML algorithm requires two kinds of dataset, which are:

Training Dataset: The algorithm uses this type of dataset to learn patterns and associations between instance features.

Test Dataset: After learning from the training dataset, the classification model is tested using this kind of dataset. In common practice, a test dataset has three times more instances than the training dataset.

4.1.3 Feature and Hyperparameter

Other than datasets, ML also requires parameter(s) as input. In fact, there are two types of parameters in ML practice, which are:

Hyperparameter: Each ML algorithm has certain parameter(s) that require explicit value(s) before the learning process takes place. These parameters(s) are called hyperparameter(s), and they have a direct effect on the ML learning output.

Feature: Feature refers to dataset attributes. For example, a dataset describing the condition of a liquid may have temperature, pressure, and volume as its features, while datasets regarding people could have feature of age, sex, address, and name.

4.1.4 Variance and Bias

Variance and bias are two terminologies that are commonly used to characterize the output of a classification model (supervised learning);

Variance: If a predictive model is said to have a high degree of variance, it means that the algorithm can learn minute detail for each class in the dataset. However, if the model variance it too high, it will memorize the details of each instance (overfitting). In this situation, the model would fail to have a generalized understanding about the difference between the classes. In practice, an overfitted model will have low accuracy in classifying new instances.

Bias: The opposite of variance is bias; it describes how an algorithm creates a generalized pattern about the difference between the classes. A biased algorithm will ignore the fine details in the sample, but emphasize more on the overall difference between the classes. Nevertheless, a model that is too biased cannot differentiate between classes at all as it ignores all the details for each class. In this situation, the model is said to be underfitting.

The objective of ML modelling is to develop a model with adequate levels of variance and bias at the same time.

4.2 Classification Algorithm

The following subchapter provides a short review of various classification algorithms, including Random Forest (RF), Support Vector Machine (SVM), Artificial Neural Network (ANN), Linear Discrimination Analysis (LDA), and Quadratic

Discrimination Analysis (QDA). Discussion about these algorithms is relevant to the research as the classification performance of each of these algorithms, except ANN, have been compared to the developed algorithm, as discussed in chapter 6.

Discussion regarding the ANN model and its usage in particle physics is given in the text. However, the algorithm itself had not been implemented in this research since vast literature is available concerning the implementation of the algorithm in particle physics.

4.2.1 Random Forest (RF)

The Decision Tree (DT) algorithm, by default, is a weak classifier; it has high variance and low bias properties. However, its accuracy can be enhanced by the 'ensemble' technique, in which multiple weak learners (such as DT) can be ensemble together to create a stronger 'committee' of classifiers. There are two popular ensemble methods; bagging (bootstrap aggregation) and boosting. The AdaBoost tree classifier is the boosting implementation of the decision tree classifier, while Random Forest is the bagging implementation (Hastie et al., 2009).

In bagging, the training dataset is broken into smaller sub-datasets and each DT is trained only with a single sub-dataset. The objective of this learning method is to create diversity among DT rather than creating a single strong classifier. With this method, the overall bias is increased compared to one individual DT (Breiman, 2001). When a new dataset needs to be classified, it can be classified based on the majority vote or by averaging the multiple DT output (Breiman, 1996).

One advantage of using RF over AdaBoost is that the bagging technique is an 'embarrassingly parallel' computing job since each individual tree that makes up the random forest can be trained independently without the need for inter-process communication. Thus, the RF model can be created in a highly parallel manner by using a machine that is multicore, multiprocess, and multithreaded. This high degree of parallelism is friendly to the CMS grid-computing system.

4.2.2 Support Vector Machine (SVM)

Support Vector Machine (SVM) is one of the most commonly used classifiers (outside the particle physics field) as it can scale to various types of data (robust) and does not require a large training dataset. The basic idea of SVM is to create a 'decision-boundary' in a mathematically-defined hyperplane, in which instances can be classified according to which side of the boundary they are located in, as illustrated in **Figure 4.1.**

The hyperplane that is needed to separate the instance according to its class may have a higher dimension than the original dataset dimension. However, creating a higher dimension is computationally costly, and the SVM algorithm employs the 'kernel trick' to solve this issue. The kernel trick is based on Mercer's theorem that avoid the need for computing at higher degree of hyperplane dimension, by just computing the inner dot product of the dataset-vector in a transformed space by using some kernel functions (Ingo & Andreas, 2008). In this research, the kernel function that was used together with SVM was the polynomial kernel function. The discussion pertaining to Mercer's theorem and kernel functions is beyond the scope of this research; interested readers are directed to the work done by Cristianini and Shawe-Taylor (2000), and Minh et al., (2006).

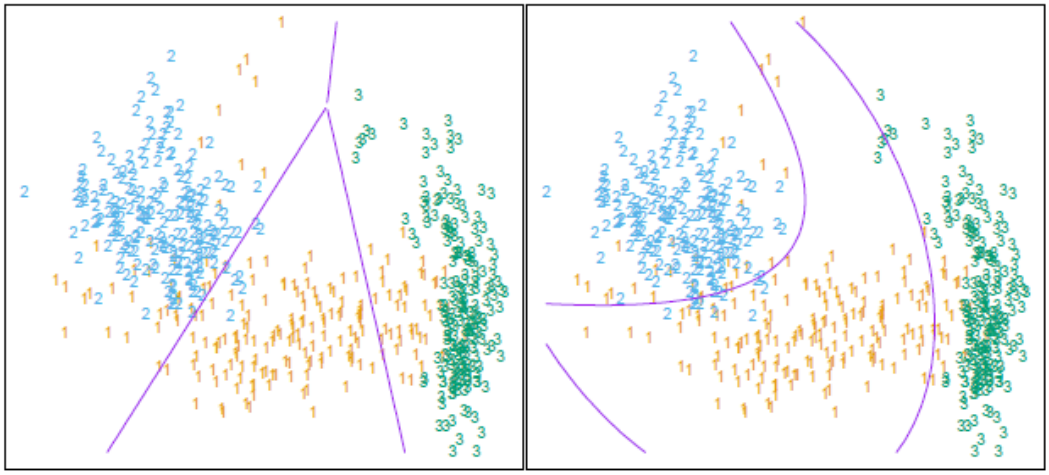


Figure 4.1: In SVM data, a decision boundary (purple line) is created in a hyperplane to give the best separation of classes. The shape of the decision boundary depends on the kernel of SVM; it can be linear (left) or non-linear (right) (taken from Hastie et al., 2009).

Moreover, it is worth noting that there are two popular versions of the SVM algorithm; the single-class SVM and the multiclass SVM. The multiclass SVM algorithm is the conventional method, in which the algorithm learns a training dataset that contains multiple labels. On the other hand, the single-class SVM learns the pattern(s) in the original dataset and predicts if a new instance belongs to the initial data set, hence detecting novelty/abnormality. Please see Khan and Madden (2010) and Yu (2003) for a more in-depth discussion regarding one-class SVM.

4.2.3 Linear and Quadratic Discrimination Analyses (LDA & QDA)

In the writing of Narsky and Porter (2013 p. 221), the Linear Discrimination Analysis (LDA) algorithm is described as:

'Because of their high interpretability, linear methods are often the first (to be used) for data analysis.'

The algorithm, which is also known as Fisher Discrimination, is commonly used in many branches of physics, including particle physics and astrophysics. Given a mixture of distributions with classes, $k = 1, 2$, the ratio of probability for an instance that belongs to each class is given as x, and the quadratic discrimination is stated in equation 4.1;

$$\log \frac{P(k=1|x)}{P(k=2|x)} = -\frac{1}{2}(\boldsymbol{\mu}_1^{\mathrm{T}}\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^{\mathrm{T}}\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2) \tag{4.1}$$

Where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the means and the covariance matrix for each class. If $\boldsymbol{\Sigma_1} = \boldsymbol{\Sigma_2}$, then it becomes a linear discriminator for the LDA, whereas if $\boldsymbol{\Sigma_1} \neq \boldsymbol{\Sigma_2}$, then it becomes a quadratic discrimination for the QDA algorithm. The hyperplane that separates the two classes can be obtained by equating the equation to zero.

In the following subchapter, example of application(s) for each classification algorithm in a particle physics analysis is given.

4.2.4 Neural Network in Particle Physics Analysis

The Artificial Neural Network (ANN) model can be classified into two type; the deep ANN and the shallow ANN. The fundamental unit in ANN is call a neuron, similar to the neuron in the human brain, the neuron will give a certain output if the input exceed certain threshold, interested reader on the subject of ANN are directed to the work of Du & Swamy, (2014).

The difference between deep ANN and shallow ANN, is that shallow ANN have fewer number of hidden layers of neuron between the input and the output layers then the deep ANN. Shallow ANN is more commonly used in the particle physics field than the deep ANN, as the it requires less computational resources.

Whiteson and Whiteson (2009) used ANN with two hidden layers to construct a top quark-Higgs event classifier. What is interesting about this research is that they used a technique call NeuroEvolution of Augmenting Topology (NEAT) to stochastically optimize their neural network, (stochastic optimization means the hyperparameter values are generated randomly until certain optimization level is obtained). The NEAT algorithm allows the neuron layer to evolve and mutate, creating new links between neurons; thus, the particular neuron linkage is not inherent to the original architecture. Furthermore, their study was conducted on a real experiment dataset collected by the Tevatron detector.

Other implementations of shallow ANN that are worth mentioning are the researchers conducted by Gupta et al., (1992) that developed a classifier between light-quark and heavy-quark events, as well as a two-tier neural network for b-tagging by Wan Abdullah (1992). Other applications of shallow ANN were by the ATLAS collaboration for its pixel detector (ATLAS collaboration, 2014) and in the search for associated production of Higgs with the top quark (The ATLAS Collaboration, 2012). Lastly, some work was carried out by Bakhet et al., (2015) to classify charged Higgs.

On the other hand, the deep ANN technique was only very recently introduced to particle physics by the pioneering work of Baldi et al., (2014). In their research, they looked into the classification of Monte Carlo-generated samples of datasets where $gg \rightarrow H^0 \rightarrow 2W + 2b$ as signal events and $gg \rightarrow t\bar{t} \rightarrow 2W + 2b$ as noise events. It was found

that the deep ANN offered better background rejection than the shallow ANN and Boosted DT, as shown in **Figure 4.2;**

In ML, a Receiver Operating Characteristic (ROC) is used to measure the performance of binary classifier. It shows the classifier sensitivity (true positive rate) as a function of its fall-out (false positive rate, i.e. false alarm). The higher the Area Under the Curve (AUC), the better the ML performance is.

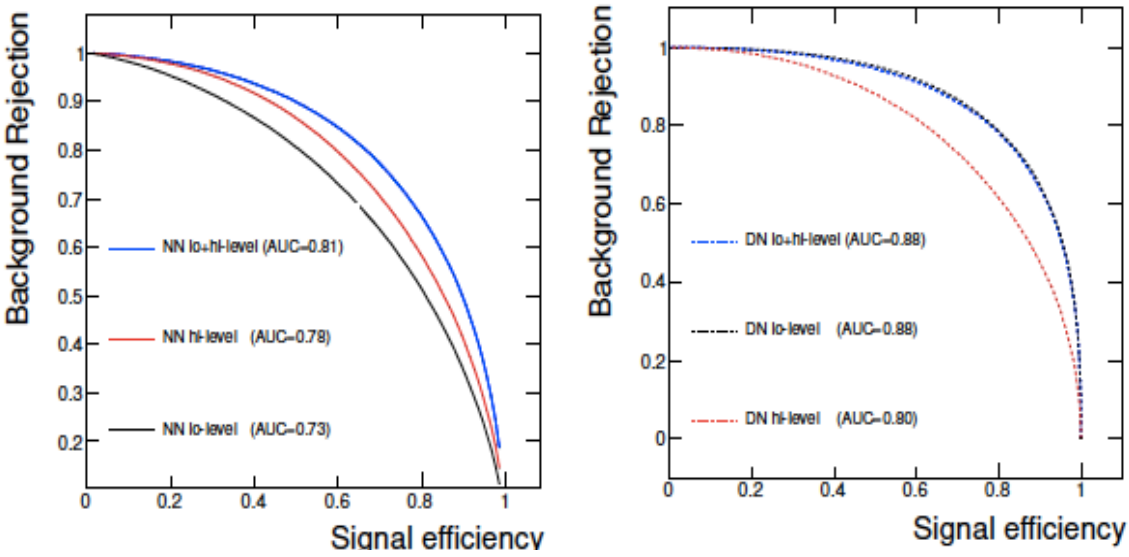


Figure 4.2: Comparison ROC between shallow ANN (right) and deep ANN (Left); the value of AUC proved that deep ANN was better the shallow ANN (taken from Baldi et al., 2014).

4.2.5 Support Vector Machine in Particle Physics Analysis

The use of SVM in particle physics field is less prevalent with notable research examples are fewer. It can be argued that SVM is a weaker classifier than ANN; however, it is more robust and requires fewer training sample than ANN.

Important research includes the work done by Vaiciulis (2002), in which the SVM was compared to other multivariate analysis methods in identifying top-quark events via the dilepton channel. The study did not find any significant difference between using SVM with the Gaussian kernel or the sigmoid kernel. The work of Vannerem et al., (1999), compared SVM to ANN as a classification tool for charm-quark tagging and muon identification for the Omni-Purpose Apparatus for LEP (OPAL) experiment. The research shows that the ANN have slightly better efficiency than SVM, as presented in **Figure 4.3**.

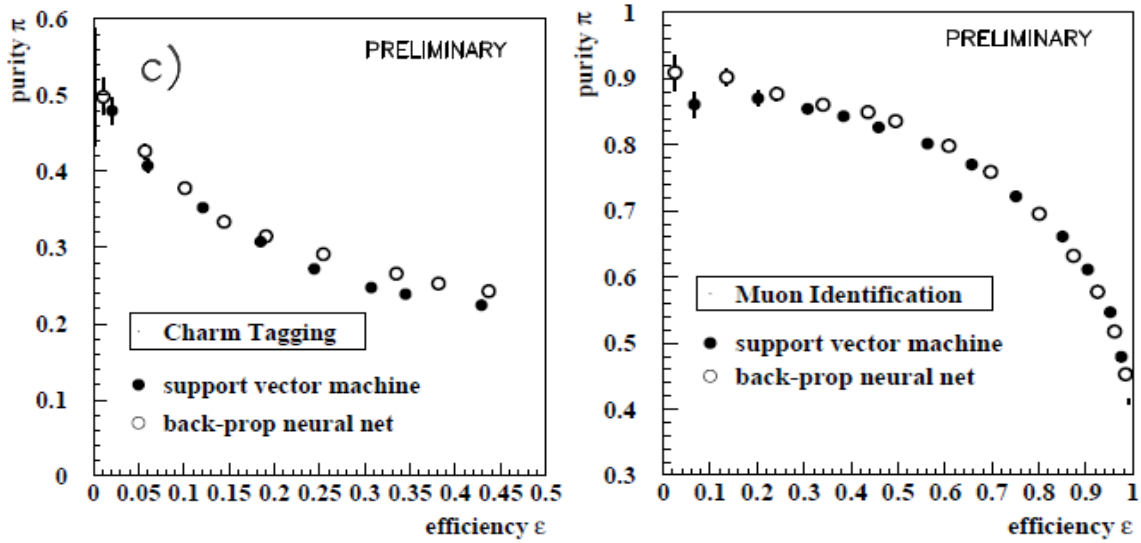


Figure 4.3: The ROC comparison between ANN and SVM for charm tagging (left) and muon identification (right) (taken from Vannerem et al., 1999).

4.2.6 Random Forest in Particle Physics Analysis

The D0 Collaboration frequently uses the random forest in its analyses; a notable one is an attempt to find the Higgs particles in the event of $H \rightarrow WW \rightarrow l\nu q'\bar{q}$. In this research, a Random Forest (RF) with 50 decision trees (a small number by today's standard) had been trained with simulated signal and background events. The

significance of this research is that the source of noise were large, including V+ jets, top quark, and diboson production (D0 Collaboration, 2011). Other examples of D0 collaboration research that used RF include the search for the Higgs boson in $ZH \rightarrow 2l^{\pm} + b\bar{b}$ (D0 Collaboration, 2012b) and the measurements of WW and WZ productions in W+ jet final state (D0 Collaboration, 2009, 2012a).

In addition, it is worth nothing that RF is also frequently used in astrophysics analysis. For example, the Major Atmospheric Gamma Imaging Cherenkov Telescope (MAGIC) and the High Energy Stereoscopic System (H.E.S.S), both employed RF in their analysis to classify events originating either from gamma rays or hadrons (MAGIC Collabration, 2007; H.E.S.S Collaborations, 2009).

4.2.8 Quadratic and Linear Discrimination Analyses in Particle Physics Analysis

In particle physics, the LDA/QDA method has been frequently used to demonstrate the superiority of multivariate classification methods over linear classification methods. For example, research conducted by Badala et al., (2008) showed that the ANN and the boosted decision tree (BDT) gave higher-purity results than the LDA method in classifying $K^{*\pm}$ event in a simulated pp collision at ALICE, as given in **Figure 4.4 (Left)**.

The superiority of ANN and BDT over LDA was again shown in the work of Heikkinen et al., (2010), where these three methods were compared in τ-event tagging for $H^{\pm} \rightarrow \tau^{\pm}\nu_{\tau} \rightarrow$ hadrons in the MSSM phenomenology, as illustrated in **Figure 4.4 (Right).**

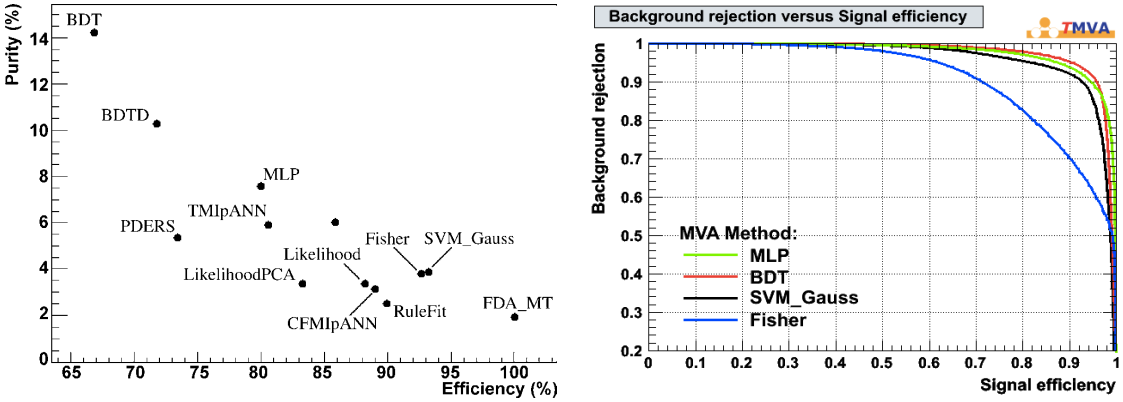


Figure 4.4: The BDT and ANN (labelled as MLP) gave higher purity results for classification than the LDA (labelled as Fisher) in classification of $K^{*\pm}$ in Badala et al., (2008) (left), a study on τ event tagging by Heikkinen et al., (2010) (right).

4.3 Clustering Algorithm Review

The term 'clustering' has been defined in several ways by various authors. However, the simplest definition is given as:

> *"Clustering is the unsupervised classification of pattern (observation, data item or feature vectors) into group."*
>
> (Jain et al., 1999, p. 264)

> *"Clustering is a mathematical technique designed for revealing classification structures in the data collected on real-world phenomena."*
>
> (Mirkin, 1997, p. 176)

Meanwhile, for more complex definition, clustering can be said as;

> *"Clustering (or cluster analysis) aims to organize a collection of data items into clusters, such that items within a cluster are more "similar" to each other than they are to items in the other clusters."*
>
> (Grira et al., 2004, p. 1)

> *"Data clustering (or just clustering), also called cluster analysis, segmentation analysis, taxonomy analysis, or unsupervised classification, is a method of creating groups of objects, or clusters, in such a way that objects in one cluster are very similar and objects in different clusters are quite distinct."*
>
> (Gan et al., 2007, p. 3)

From these various definitions, it can be said that the central idea of clustering is to find an underlying structure or pattern for a given dataset (Backer & Jain, 1981), which can be a form of summary of the dataset (Fahad et al., 2014). Clustering methods are vital to present day scientific research, in which high-dimensional data are generated at an exponential rate. Thus, a clustering action transforms a very complex dataset to a summary of patterns that is more understandable.

From the various definitions provided, it can be concluded that clustering has three main objectives, as listed in **Figure 4.5**;

Clustering	**Underlying structure**: To gain insight into data, generate hypotheses, detect anomalies, and identify salient features
	Natural classification: To identify the degree of similarity among forms or organisms (phylogenetic relationship) and to assist in classification design
	Compression: As a method for organizing the data and summarizing it through cluster prototypes

Figure 4.5: The three main objectives of clustering (Jain, 2010; Mirkin, 1997)

Clustering methods are also heavily used in many areas of studies, such as computer vision, and data mining; a comprehensive survey can be found in Jain et al., (1999) and Jain and Dubes (1988).

4.3.1 Clustering Algorithm

The most conventional clustering algorithm is undoubtedly the K-Means clustering. The algorithm has been proven to be robust and fast in clustering datasets. Let $X = \{x_i\}, i = 1, \dots, n$ be the set of points with n dimension of feature to be clustered in to K number of cluster, $C = \{c_k, k = 1, \dots, K\}$. The K-Means algorithm works by minimizing the sum of the squared error between an instance and the empirical means, $\boldsymbol{\mu}_k$ of the cluster, given in equation (4.2) (Jain, 2010);

$$J(C) = \sum_{K=1}^{K} \sum_{x_i \in C_k} \|\boldsymbol{x_i} - \boldsymbol{\mu}_k\|^2 \tag{4.2}$$

However, the difference between instance vector, $\boldsymbol{x_i}$ and the hypothesize cluster mean, $\boldsymbol{\mu_k}$ reduces as the dimension increases. Thus, the K-Means algorithm is known to be a poor classifier for high-dimensional datasets and tends to generate a cluster with a normal distribution. The reduction in measurable dissimilarity among instance as the dimension increase is known as the 'Curse of Dimensionality' and is discussed in detail in subchapter 5.3.

Over the years, numerous extensions of K-means have been developed to better adapt real world data, including:

- Fuzzy c-means (Bezdek, Ehrlich, & Full, 1984; Dunn, 1973)
- Bisecting K-means (Steinbach, Karypis, & Kumar, 2000)
- Kd-tree (Pelleg, & Moore, 1999)
- X-means (Pelleg, Pelleg, Moore, & Moore, 2000)
- Kernel K-means (Schölkopf, Smola, & Müller, 1998)
- K-medoids (Kaufman, & Rousseeuw, 1987)

In this research, another extended version of K-Means, also known as the Dirichlet Process Gaussian Mixture Modelling (DPGMM), had been used.

4.3.2 Dirichlet Process Gaussians Mixture Modelling Algorithm

Equation (4.2) shows that the K-mean algorithm requires users to hypothesize the number of cluster K, before clustering. However, in practical application the value of K cannot be known precisely.

Hence, the Dirichlet Process Gaussian Mixture Modelling (DPGMM) extends the K-Means algorithm so that it no longer requires the K hyperparameter to be specified. It introduces a new hyperparameter alpha, α, which is the instance density in a given cluster. In this research, the DPGMM was used to cluster a trained SOM centroid in as will be discussed in Chapter 7. However, it is important to note that DPGMM still exhibit the low clustering capability for high-dimensional dataset as K-Means algorithm (Markou, & Singh, 2003). The theoretical background for DPGMM is beyond the scope of this study; however, interested readers are directed to the writing of Görür and Edward (2010).

4.3.4 Clustering in Particle Physics

Algorithms, such as kt, anti-kt, and Cambridge/Aachen, which are used for jet reconstruction (Cacciari, Salam, & Soyez, 2008), can be said to be a form of clustering algorithm. These algorithms are strongly infused with the particle physics knowledge; making it a 'domain-specific' algorithm and impractical to be used outside the field.

Furthermore, to the best of author's knowledge, there is no strong practice in using a non-physics domain-specific clustering algorithm in particle physics analysis. A rare example is the study done by Chekanov (2006), where the K-Mean algorithm was employed for jet reconstruction from heavy particle events.

The aim of this research is to study whether it is practical to use a non-physics domain-specific clustering algorithm such as the SOM and the DPGMM in analysing particle physics dataset.

4.4 Introduction to Self-Organizing Map

The Self-Organizing Map (SOM) is the core of this research. This clustering algorithm has been used in several scientific disciplines, including signal processing, computational network analysis, and genetic study (Honkanen, Liuti, Carnahan, Loitiere, & Reynolds, 2009).

Teuvo Kohonen developed the algorithm in 1982, taking inspiration from the human brain which consist of different areas carrying out different cognitive functions. He further described the algorithm as (Kohonen, 1982, 2013);

> *'a projection mapping similar to vector quantization with the addition of being 'spatially-globally ordered'.*

In most literature, the SOM is categorized under ANN-based algorithm (Acat, & Heikkinen, 2007; Dittenbach, Merkl, & Rauber, 2000; Gan et al., 2007; Jain et al., 1999; Lange, Hermanoski, & Freiesleben, 1997). However, the author is more inclined to the view of Fahad et al., (2014), who classified it as a modelling-based algorithm, as presented in **Figure 4.6**. Description of each clustering algorithm stated in **Figure 4.6** can be found in Fahad et al., (2014).

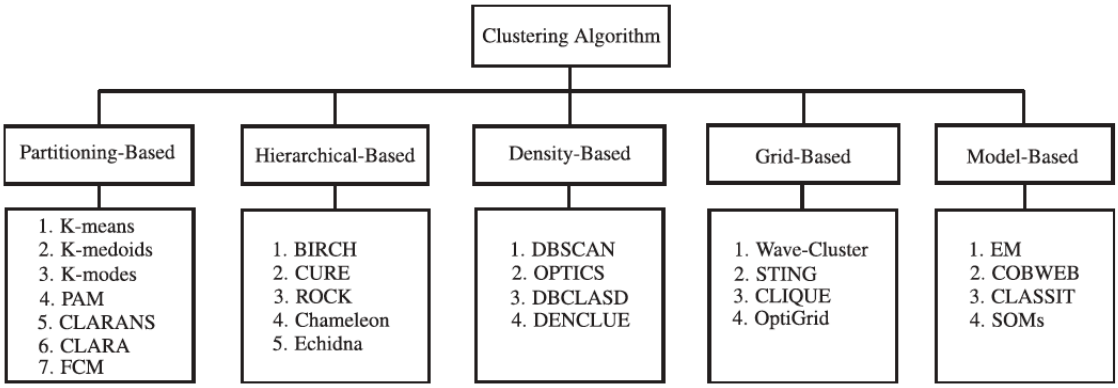


Figure 4.6: The classification of different clustering algorithms by Fahad et al., (2014).

The different view of which category the SOM algorithm should be classified in is caused by the fact that the algorithm is composed of smaller learning units similar to ANN, as illustrated in **Figure 4.7**. For ANN, the fundamental learning unit is called a 'neuron', and several authors have used the same terminology in naming the SOM learning unit. However, the SOM and the ANN learning units are very different to one another, both in terms of learning mechanics and learning output.

Since the SOM learning unit does not learn the same way as a neuron, the author used the label 'centroid' to name the SOM learning unit, analogous to the naming convention for the basic unit of the K-Means clustering algorithm. The reason for this naming convention is provided in the following subchapter.

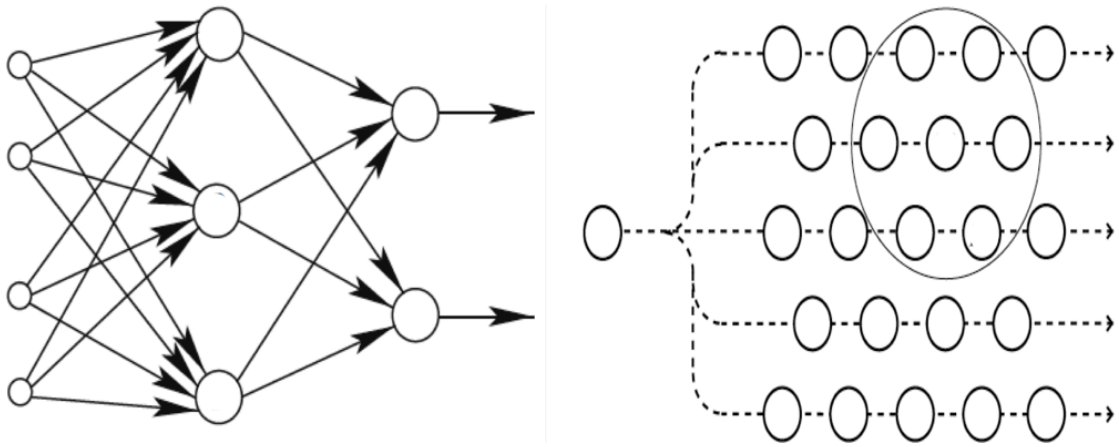


Figure 4.7: General depiction of neural network (Right) and SOM (left) fundamental units. For a neural network, this unit is called neuron, while the SOM unit is called centroid. Image Source: Du & Swamy (2014), and Kohonen (2013), respectively.

4.4.1 How the SOM Model Learns?

Creating a SOM model starts with creating a map of centroids, which is called a feature-map or SOM map, where the common map is a 2-dimensional plane with the centroids having equal distance to one another, as shown in **Figure 4.8**. Each centroid has a position (x, y) and a vector called a weight-vector ($\boldsymbol{x_c}$). The initiation process also transforms every instance in the dataset into a vector notation, (4.3) which allows all instances to be treated as vectors.

$$x = (x_1, x_2, \dots x_n) \rightarrow \boldsymbol{x_i} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \dots \boldsymbol{x_n}) \tag{4.3}$$

The SOM model training starts by randomly selecting an instance (in a vector format) from the training dataset. Then, the similarity between this instance and each centroid weight-vector is determined and measured by a given '*similarity function*'. The centroid with the least difference is chosen as the winning centroid. The next action is to

perturb the weight-vector for the winning centroid and all centroids with a smaller distance to the winning centroid than R_{max}, with equation (4.4).

$$\boldsymbol{x}_{c,n+1} = \begin{cases} \boldsymbol{x}_{c,n} + l(t) \cdot \dfrac{R_{max} - r_{n,w}}{R_{max}} \cdot (\boldsymbol{x_i} - \boldsymbol{x}_{c,n}), & r_{n,w} \leq R_{max} \\ \boldsymbol{x}_{c,n}, & r_{n,w} > R_{max} \end{cases} \tag{4.4}$$

$$R_{max} = D(t) \tag{4.5}$$

where $l(t)$ and $D(t)$ are the learning-rate function and radius-decay function respectively. Commonly, the learning-rate function and the radius-decay function are defined as a decay function over the training iteration, t. Thus, the learning-rate value and the maximum radius R_{max} will decay over the course of the training, creating a convergence on the centroid weight-vector value.

The process of randomly selecting an instance, followed by determining the winning centroid and then perturbing the winning centroid, as well as the selected centroids, is considered as one (training) iteration. This iteration is repeated until the training phase ends. When the training phase is completed, an SOM model has been created.

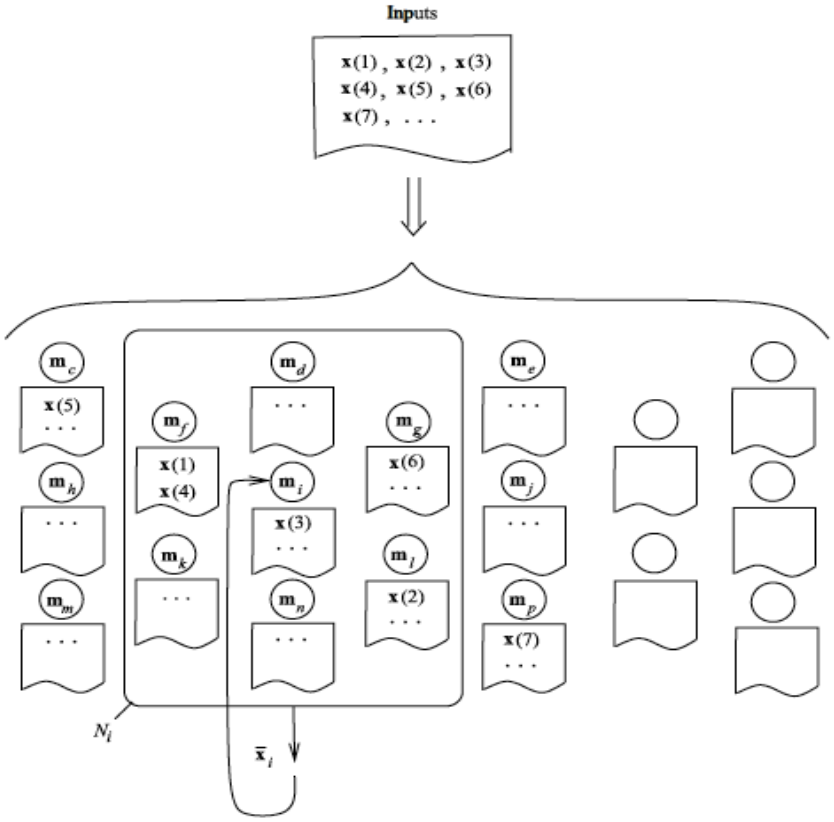


Figure 4.8: Each circle denotes a centroid, while the square is the collection of centroids with a distance from the winning centroid (m_i), less than $\boldsymbol{R_{max}}$

From the explanation given earlier, it can be argued that the SOM learning method is distinct from those practiced in ANN modelling. ANN model is developed from a supervised learning method, whereas the SOM model is generated via the unsupervised learning method.

4.4.2 Example of SOM in Particle Physics Research

Compared to other ML algorithms, the implementation of SOM in particle physics is limited, with most SOM models used predominantly for improving background-event rejection (Honkanen et al., 2009). Two examples refer to the study conducted by Lange, Hermanoski, and Freiesleben (1997), as well as Lange, Fukunaga, Tanaka, and Bozek,

(1999); in the 1997 publication, the SOM model was used in the COSY-TOF experiment for background-event rejection for pion analysis in a pp collision ($pp \rightarrow pn\pi^+, pp \rightarrow pp\pi^0, pp \rightarrow d\pi^+$). Meanwhile, the 1999 publication mentions the use of the SOM technique as an event filter for the BELLE experiment. What is interesting about both publications is that they used a modified version of SOM, which included a novel concept called 'node-gravity'.

In the original SOM algorithm, which is equation (4.3), there are two free variables known as the learning-rate and the radius-decay function. In these two publications, the concept of node-gravity simply combines these two parameters into one single parameter, α, and describes it as the gravitational pull strength between the centroids (see Lange and Freiesleben (1996) for further details).

In this research, the concept of node-gravitational pull was not implemented since the two functions in the original equation of (4.3) had been perceived to be a positive attribute of SOM as it allows the equation to be flexible and adaptive to multiple data types.

It is worth mentioning here that Tryba and Goser (1991) modified the SOM equation with the Schrodinger equation. Among the few SOM usages in particle physics research, one of the most exceptional is the work of by Honkanen et al., (2009). While other studies used SOM for background discrimination analysis, this research employed SOM as a stochastic optimizer in selection/fitting of Parton Distribution Function (PDF). No modification was done on the SOM algorithm in this research, except that the author used an SOM training mode called 'batch-mode'. In batch mode, the centroid weight-vector is not perturbed in an iterative training method, but it is done in a single operation,

giving a faster computational execution. However, the 'batch-mode' training was not used in this research as insufficient researches are available to validate that both batch mode and iterative methods could produce the same model. Especially for SOM model that is generated from unconventional learning-rate function and similarity matrix.

4.5 Development of Self-Organizing Map

One of the objectives of this research had been to develop an application that creates an SOM model for clustering and classification of particle physics events. The development of the application had to be done from scratch as other SOM implementations were not designed to process big datasets common in particle physics. Even though this research also used various other ML algorithms, only the code for SOM was developed by the author. All other ML codes were taken from another ML module library.

The SOM code was fully written in the Python language (version 2.7) with several non-standard libraries, including scipy, numpy, scikit, pyplot, and pandas. Scipy and numpy are Python modules that are developed for scientific research, thus provide the necessary code to do statistical processing, statistical analysis, and vector manipulation. Meanwhile, the pandas module plays a crucial role in the implementation as it provides the necessary code to manipulate datasets in a datasheet object that dramatically reduces the execution runtime. Datasheet object is an object in the python language that act as a datasheet. Other than that, pyplot, as its name suggests, provides the module for data visualization.

Scikit, on the other hand, is a Python ML library that provides the clustering and classification API. Details regarding this module can be found in Pedregosa et al., (2012). In this research, this module had been heavily used for a dataset pre-processing, as well as for the classification algorithm implementation. The development of the SOM code is comprised of three stages; initiation phase, training phase, and post-processing phase.

4.5.1 Initiation Phase

Two processes have to be carried out during the initiation phase, dataset normalization and SOM feature-map initialization. Normalization is important for any ML algorithm that uses similarity functions, as it maps the different value range between features to a scale between 1 and -1. Without it, feature(s) with a larger magnitude (or range) will have a more dominant affect upon the similarity measurement than instances with lower magnitude.

In fact, there are two common normalization methods; Z normalization (equation 4.5) and min-max normalization (equation 4.6). Only Z normalization is implemented in the code as it only requires the mean, μ and the standard deviation, σ from the original dataset (training dataset). x is the instance that is required to be normalize, while x_{min} and x_{max} are the minimum and maximum values in a given dataset. z and x' are the normalized value using the Z normalization and min-max method respectively

$$z = \frac{x - \mu}{\sigma} \tag{4.5}$$

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.6}$$

The second step is to initialise the SOM feature-map itself; this requires the number, the topology, and the initial weight-vector of the centroid to be defined. In this research, the most common configuration for the feature-map is defined to be a 30 × 30 square-shaped map, as shown in **Figure 4.9 (left),** while the SOM model for the dimuon dataset is set at a size of 100 × 100.

The initial weight-vector for each centroid is also assigned randomly. The randomization creates asymmetry distribution across the SOM map, as depicted in **Figure 4.9 (right)**, as well as diversity in the weight-vector among centroids. These ensure that the SOM algorithm will construct a diverse similarity matrix (matrix that is constructed by a similarity function on multiple instances or points) during the training phase.

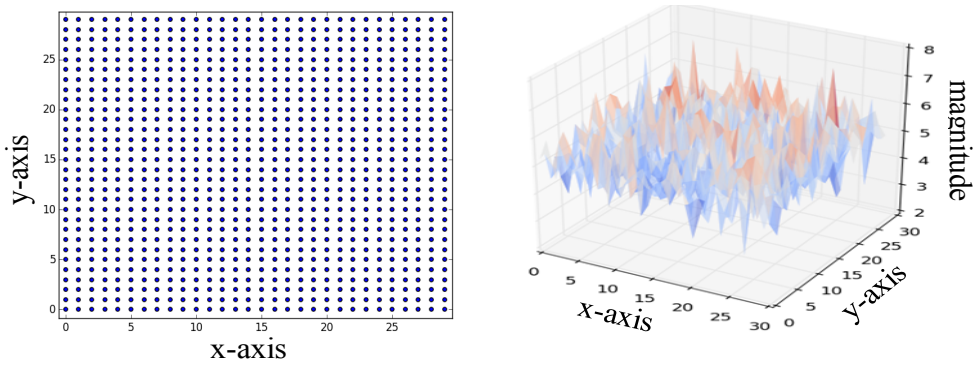


Figure 4.9: (Left) SOM centroids with 30 × 30 square shape distribution, (Right) randomized initial centroid where Z-axis is the value of the centroid weight-vector.

4.5.2 Training Phase

The implementation of the training phase is same as described in subchapter 4.4.1, and summarized in **Table 4.1**;

There are several functions in the training phase that require mathematical definition: similarity function (to create the similarity matrix), the learning-rate function, and also the radius-decay function. There are various methods for calculating the similarity between vectors; the conventional SOM implementation uses the Euclidean distance (Kohonen, 2013). Nevertheless, in this research, several other similarity measurement methods had been used to create the similarity matrix, which are explained in detail in Chapter 5.

Table 4.1: The implemented training phase for SOM algorithm

For t ≤ max training iteration:
Randomly choose instances vector from the training dataset
Create similarity matrix between instance vector and centroid weight-vector
Most similar centroid is the winning centroid
Calculate $r_{n,w}$ for the current training iteration
Select centroid with distance to the winning centroid less than $r_{n,w}$
Calculate the current learning-function, $l(t)$ for the current training iteration
Perturb winning and selected centroids with equation 4.3

As stated earlier, $l(t)$ and $D(t)$ have been commonly taken as a decay function over the training iteration, t. The learning-rate and the radius-decay function play a significant role in the SOM modelling as they;

1. create convergence on the centroid value during training, and
2. create a globally ordered centroid distribution.

The creation of globally ordered centroid can be seen in **Figure 4.10**, in which the initial (randomized) distribution of the centroid weight-vector becomes 'smoother' relative to its neighbour as the training iteration increases. This smoothing effect is attributed to the learning-rate function; thus, it is also referred to as the 'smoothing function'. The importance of a globally ordered distribution is that it forces the centroid weight-vector to be approximately similar to its neighbour, as shown in **Figure 4.11**.

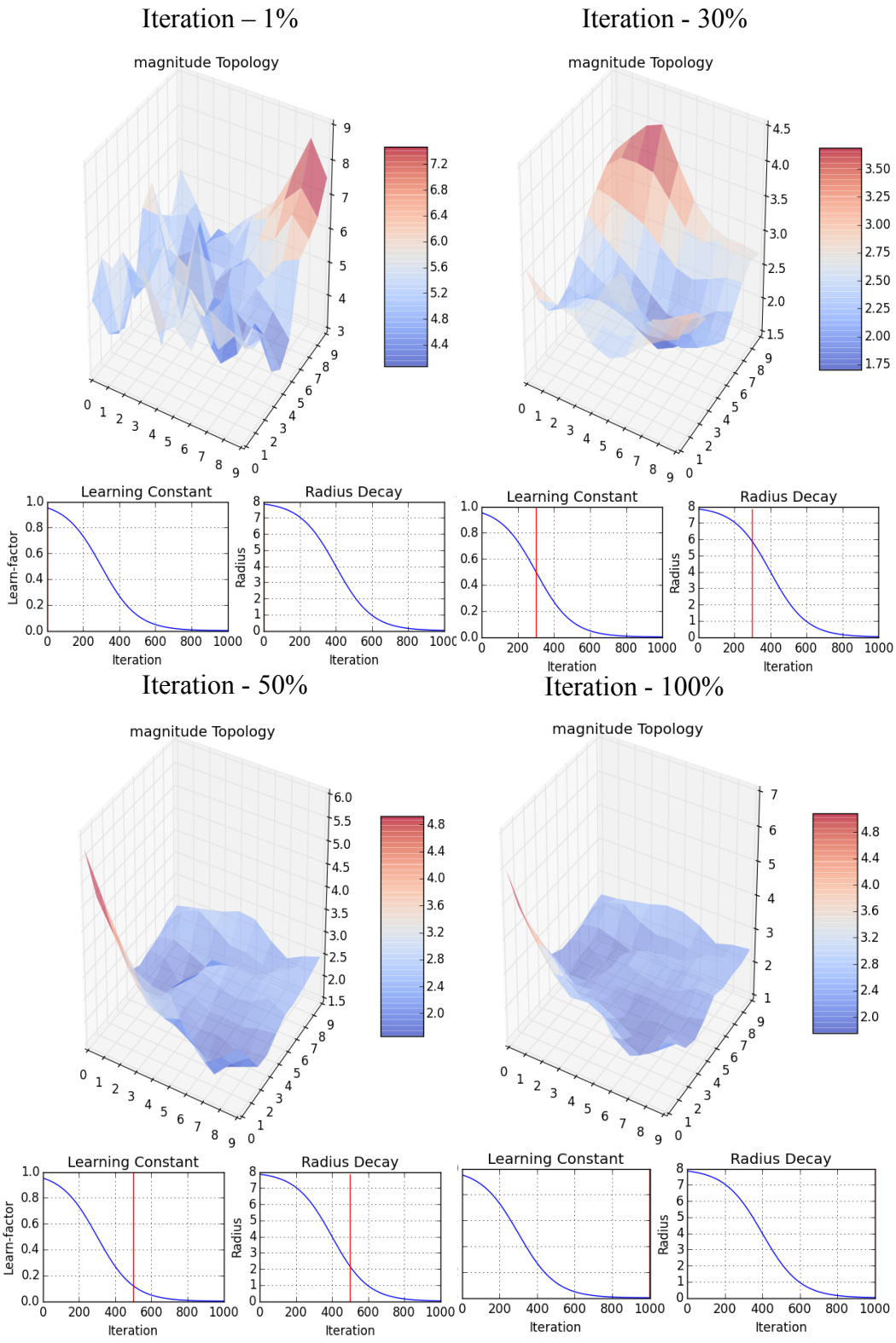


Figure 4.10: The evolution of the centroid weight-vector magnitude across the SOM map from 1%, 30%, 50%, and 100% of the maximum number of training iteration.

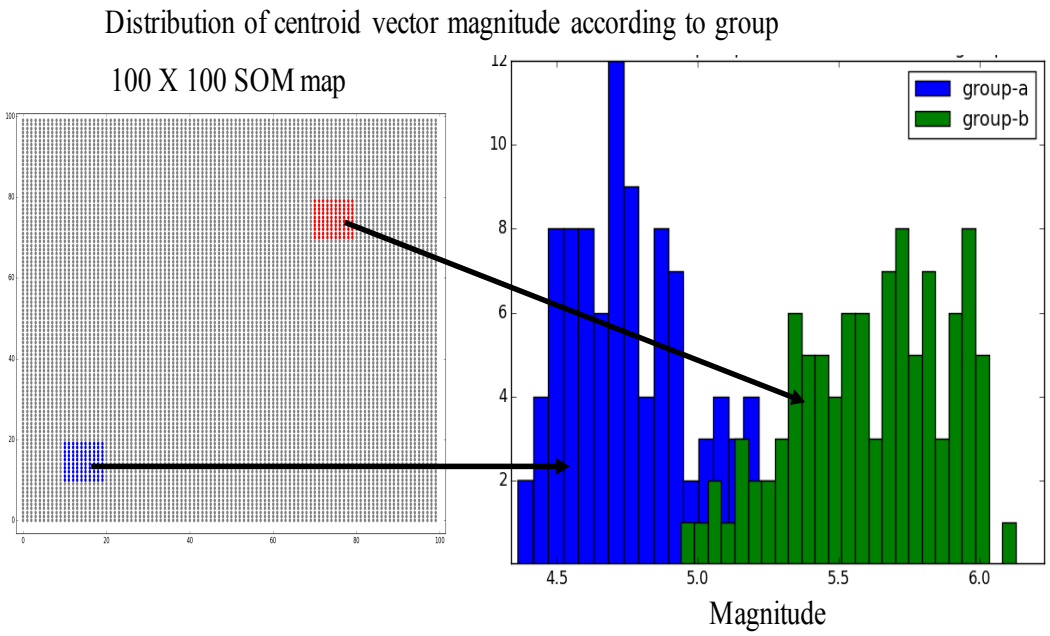


Figure 4.11: Centroids that are close to each other (group) have more similar.

The exact mathematical equation for $l(t)$ and $D(t)$ has never been discussed in the literature, thus it is open to interpretation. The effect of different learning-rate function on the SOM model is done in the next chapter.

4.5.3 Mapping Phase

After the training phase is completed, mapping of instances to the SOM map can be done by pairing each instance to the centroid with a weight-vector most similar to the instance vector. After this process, a group of instances (known as Local Instance Cluster, LIC) is created for the majority of the centroids whose instances share similar values. A very small number of centroids will not have any LIC as no instances is mapped to them. For example, in **Figure 4.12**, centroid C_1 has a weigh vector that is almost similar to instances X_a, X_b, and X_c vector. These three instances are then mapped to centroid C_1,

forming C_1-LIC. Likewise, instances X_d, X_e, and $X_{f\ are}$ mapped to centroid C_2 and form C_2-LIC.

Besides, **Figure 4.11** shows that the neighbouring centroid has the tendency to share similar weight-vector. Thus, it can be said that neighbouring centroids also form a cluster of centroids with similar weigh vector. Multiple LIC in the centroid cluster form a Global Instance Cluster (GIC), as presented in **Figure 4.12**.

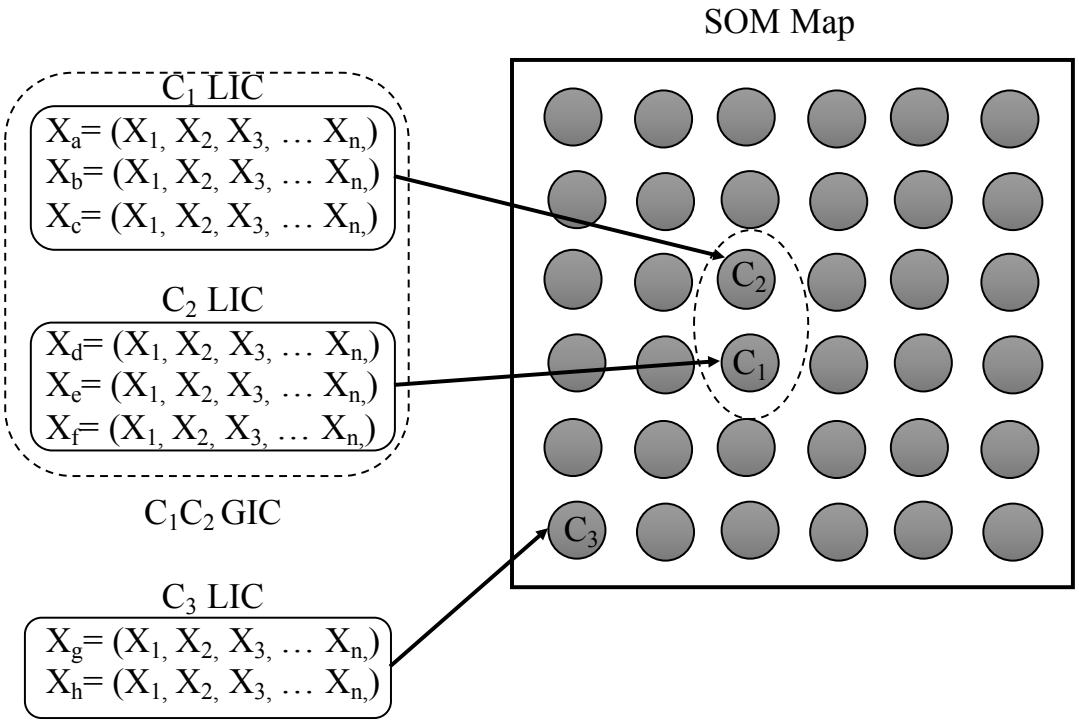


Figure 4.12: Instances X_a, X_b, and X_c have a vector similar to the C_1 weight-vector and form C_1-LIC. Likewise, X_u, X_u, and X_u form the C_2-LIC. Since centroids C_1 and C_2 are similar, their instances collectively form C_1C_2 –GIC. However, centroid C_3 does not have a weight-vector similar to C_1 and C_2, and thus its instances do not belong to C_1C_2 –GIC.

4.5.4 Post-Processing Phase

After the training and the mapping phases are completed, the SOM model is saved. It is worth noting here that saving the model is essential since there are two random processes in the SOM model creation. The first randomization occurs in the centroid weigh-vector initial value, while the second randomization occurs in selecting instances during the training phase. Thus, even if two SOM model are created using the same dataset and configuration, these two models will not be exactly the same due to these two randomize processes.

In this implementation, the initial training condition, as well as the final SOM output, is saved in the format of comma-separated-values (.csv) since this format is readily readable by other software programs. The other process that occurs during the post-processing is the data visualization process, where various histograms, 3D-Plots, and contour plots are made for validation and analysis purposes.

The complete code architecture of SOM processing included the post-processing phase is given in Appendix A.3.

4.5.5 SOM Development Conclusion

The self-organizing map (SOM) is a clustering algorithm that learns through unsupervised method. Thus, an application based on the SOM algorithm for clustering particle physics instances have been developed. The application comprised of four phases; initiation, training, mapping, and post-processing. The SOM model was

developed by first normalizing the dataset and then creating the centroid map (SOM map) in the initialization phase. Afterwards, each of the centroid's weight-vector was perturbed in the training phase with equation (4.4). Once the training phase had been completed, the dataset instances were mapped to the SOM map in the mapping phase. The post-processing phase ended the process by providing visualization and analysis of the completed model.

4.6 Parallelization

The SOM modelling runtime increases sharply as the number of centroids increases (Cuadros-Vargas et al., 2003; Fahad et al., 2014); even for a small-sized SOM map, the model would require higher computing time than most clustering algorithms. One way to decrease the computation time is by executing the training phase in a parallel manner. Relevant literature regarding SOM model parallelism has been written by Seiffert and Jain (2002).

In fact, several methods are available for transforming a recursive computational execution into a parallel execution, however in this research, the multiprocessing solution was selected. The benefits of the multiprocessing approach include:

1. The SOM training phase requires a large memory overhead therefore, it is not suitable for the multithreading technique.
2. The application is built by using the Python language, which discourages the use of multithreading since it uses the ‘global interpreter lock’ upon variable assignment.
3. The multiprocessor architecture allows it to be easily scaled up to a multi machine design easily.

To achieve this high-level of computing efficiency in the multiprocessor method, it is necessary to make each sub-process as independent as possible from the main process and to minimize the need for inter-process communication. For these condition to be satisfied, the centroids developed during the initiation phase have to be divided equally between the sub-processes. Then, the values for the learning-rate function and the radius-decay function are pre-computed and given to all sub-processes before the training phase starts. Now each sub-processes have enough information to run nearly independently in the training phase. After the training is completed, all the centroids are joined back together, forming a single SOM map.

Nevertheless, this is not an 'embarrassingly parallel' method as process are not fully independent to one another. In each training iteration, the child-process is required to communicate with the main process to give the updated similarity matrix and then obtain the position of the new winning centroid. All sub-processes are idle during this step, as depicted in **Figure 4.13**.

The requirement for inter-process communication during the training phase is a bottleneck for the current implementation. It also places a limitation on the number of sub-process that can be spawned during any training iteration. If the number of sub-process increases beyond this limit, the total execution time will increase sharply as the intercommunication time between the sub-processes will be longer.

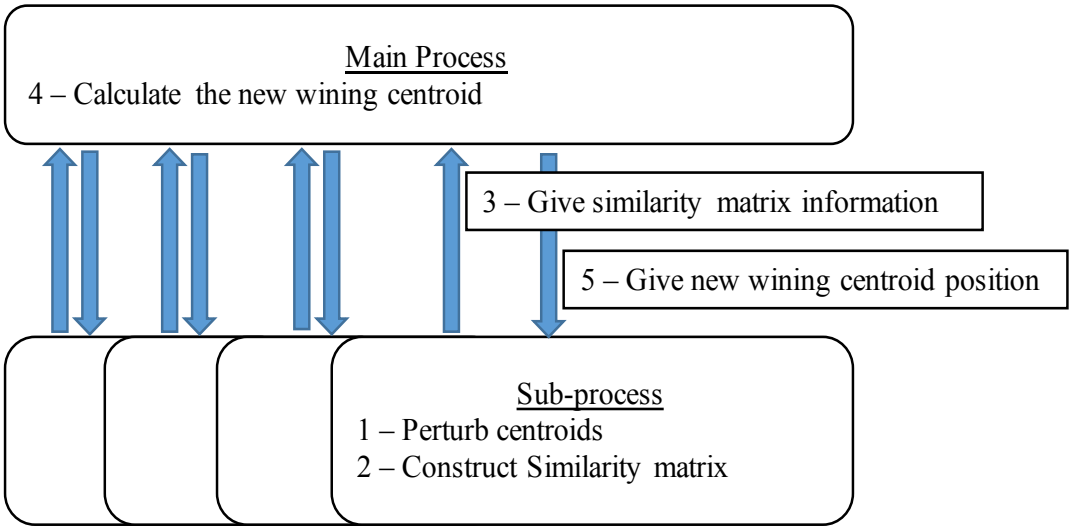


Figure 4.13: The intercommunication between the main and the sub-processes in the training phase, where all sub-processes are idle after step 3 until the next iterative.

4.6.1 Training On the Cloud

Keeping the number of sub-process below the limit means that an individual sub-process has a higher computational workload, which in return, requires a larger memory allocation for each sub-process. In certain cases, one physical machine (computer) would not contain enough memory to support all the sub-process memory requirements, thus, the SOM modelling process has to be extended beyond one machine.

As the training process is fragmented to run on multiple CPUs concurrently, the actual physical locations of the CPUs are independent of the main process physical location. Whether all the CPUs are on a single physical computer or scattered across the globe, the training and the mapping processes still can be executed in the same manner.

Moreover, the advance of cloud computing in recent time has allowed the deployment of new server(s) anywhere in the world easily. Besides, leveraging the cloud-based solution permits the scaling up of the SOM training phase into a multi machine process at the global scale. Hence, in this scenario, the amount of memory (or CPU) available for the SOM modelling is no longer bounded to the computing resources in a single.

Another advantage of using cloud computing for SOM model training is that the overall performance would not degrade considerably upon executing the training on a virtual-CPU (vCPU). The reason for this is that in every training iteration, there will always be a period when the CPU is idle (**Figure 4.13**). Unutilized CPU time can be used by another vCPU that is mapped to the same physical CPU.

4.6.2 Cloud-Based SOM Prototype

From the development point of view, the most obvious solution for up-scaling the SOM training beyond one physical machine is to use a computer cluster architecture rather than cloud-based computing. Nevertheless, there is little difference between a cloud-based solution and computing cluster in terms of system architecture. Inter-node communication in a computing-cluster will usually use the Message Passing Interface (MPI), while the inter-node communication in cloud can be handled by using the Transmission Control Protocol (TCP), however, the message that is being transmitted remains the same. Thus, the only substantial differences between these two systems are the network latency and the stability. An application that can be deployed in the cloud can be easily ported to also work in a computing cluster, such as UM-sifir.

Therefore, a rapid prototype of a cloud-based SOM algorithm was developed to study the practicality of running such an algorithm. The network latency and the stability were measured in terms of idle time, which referred to the period spent by the worker server in idle in waiting for the master server respond. The Amazon Elastic Cloud Compute (EC2) environment was chosen as the cloud infrastructure as it allowed a virtual server to be deployed at different countries across the globe.

4.6.3 Cloud vs CRAB3

Before going through the cloud implementation, is it worth noting that the CMS grid computing could not be used for multi-machine SOM modelling. The CRAB3 that had been developed for the CMS collaboration does allow personal algorithms to be run on its grid computing infrastructure. However, in a conventional grid architecture, a sub-process (in grid-computing, it is called a task) does not have the authority to communicate with other processes (sub or main). Without inter-process communication, the training process, as shown in **Figure 4.13,** could not be performed, thus the SOM model could not be created in the CMS grid environment.

Nevertheless, cloud computing has always been part of the CMS computing ecosystem, acting as an opportunistic computing resources. For example, the work done by Evans et al., (2011) showed that the EC2 infrastructure could be used to provide additional temporary resource when there was a spike in usage. Additionally, Hufnagel (2015) stated that the CMS Tier-0 would also be ported to the CERN internal cloud computing infrastructure. Thus, cloud computing will have a more significant role in the CMS computing ecosystem in the near future.

The SOM mapping phase is an embarrassingly parallel job, which decouples the need for interconnection between servers and it can be executed on the CMS grid infrastructure. However, the mapping of large instances to the SOM model can be executed in a very short period of time in a single-multicore machine; less than 15 minutes for 15,000 instances. Thus, developing and executing such task on the CMS grid is deemed unnecessary and was not pursued in this thesis.

4.6.4 Prototype Implementation

The cloud implementation only requires the pre-processing phase and the training phase to be redeveloped to match a cloud framework. In this framework, there was no sub/main process, since all processes are a main process in nature. However, the role of the machine (cloud servers) was divided between 'master' and 'worker' machines.

The master server role handled the inter-communion between all worker servers, besides determining the winning centroid in each training iteration (similar to the main process). The worker servers were designed to perturb the centroid weight-vector and to construct a similarity matrix (similar to the sub-process task). The physical locations of these virtual servers were in Universiti Malaya, Singapore, Sydney, Frankfurt, Sao Paulo, Ireland, and California, as shown in **Figure 4.14**. All servers, except that at Uni. Malaya, were EC2 servers and in each location, only one server was deployed:

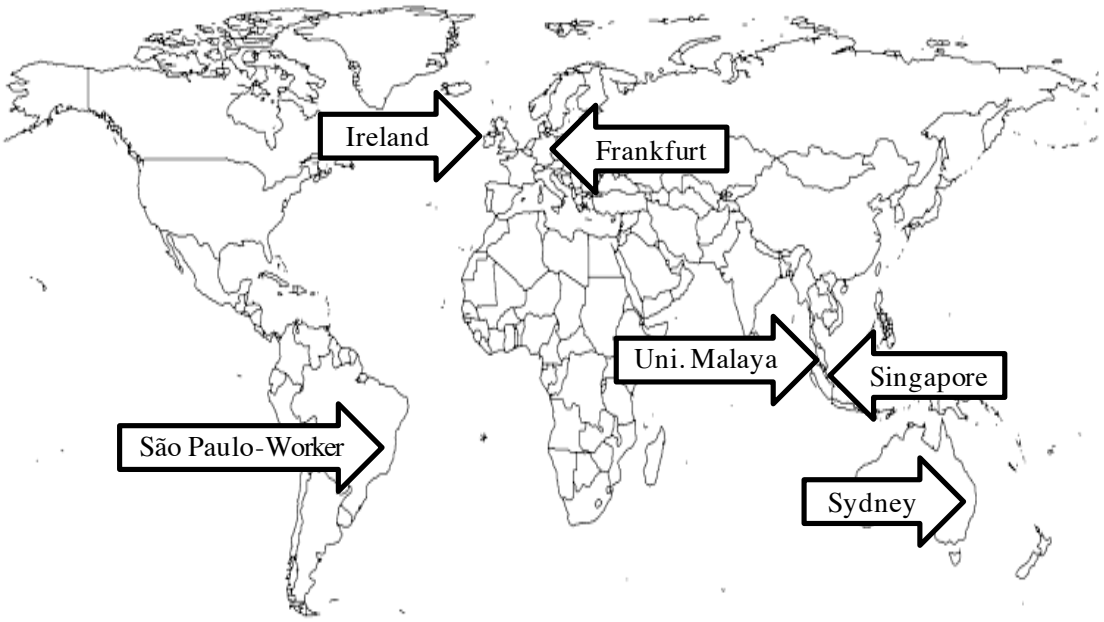


Figure 4.14: Physical location of the servers across the globe

Furthermore, in order to test network stability and latency, the idle time was recorded under the following configurations:

1. One-to-one connection with the master server in Uni. Malaya, while the worker server was in California
2. One-to-one connection with the master server in Uni. Malaya, while the worker server was in Frankfurt
3. One-to-one connection with the master server in Frankfurt, while the worker server was in California
4. One-to-one connection with the master server in Singapore, while the worker server was in Frankfurt
5. One-to-many connections (5) with the master server in Uni. Malaya, while the worker servers were in California, Ireland, Singapore, Sydney, and Frankfurt

4.6.5 Recorded Idle Time

Figure 4.15 and **Table 4.2** show the characteristics of the idle time recorded for one-to-one connection between the master and the worker servers located in different cities;

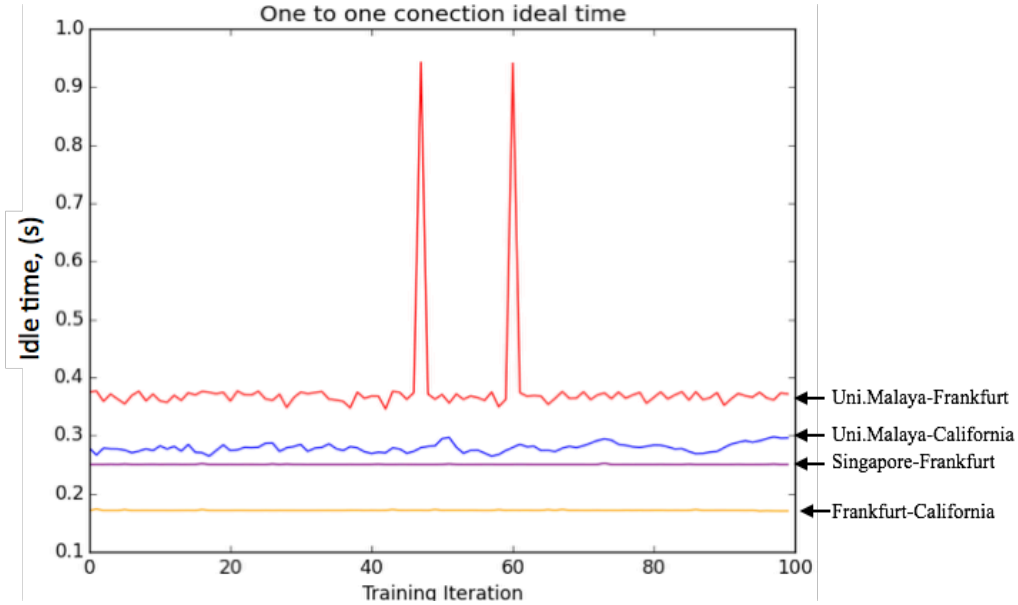


Figure 4.15: The idle time for one-to-one connection based on master and worker server locations.

Table 4.2: Idle time mean and standard deviation (STD) according to server locations for one-to-one connection.

Server Location	Idle Time (s)	
	Mean	STD
Frankfurt-California	0.1714	0.0006
Singapore-Frankfurt	0.2500	0.0004
Uni.Malaya-California	0.2789	0.0075
Uni.Malaya-Frankfurt	0.3784	0.0812

Figure 4.15 and **Table 4.2** show that the location of the server had a very strong effect on the idle time, as consequence of the training period for the SOM model. Connection to/from Malaysia displayed a tendency to have a higher variance (Uni.Malaya-Frankfurt had an STD of 135 times from that of Frankfurt-California) and a longer idle time. Thus, if a Cloud-based SOM were to be deployed in production, it should be done outside of Malaysia to ensure lower idle time. Other than that, in terms of multiconnection, **Figure 4.16** and **Table 4.3** present the characteristics of idle time for the worker-server locations with the master server located at Uni. Malaya;

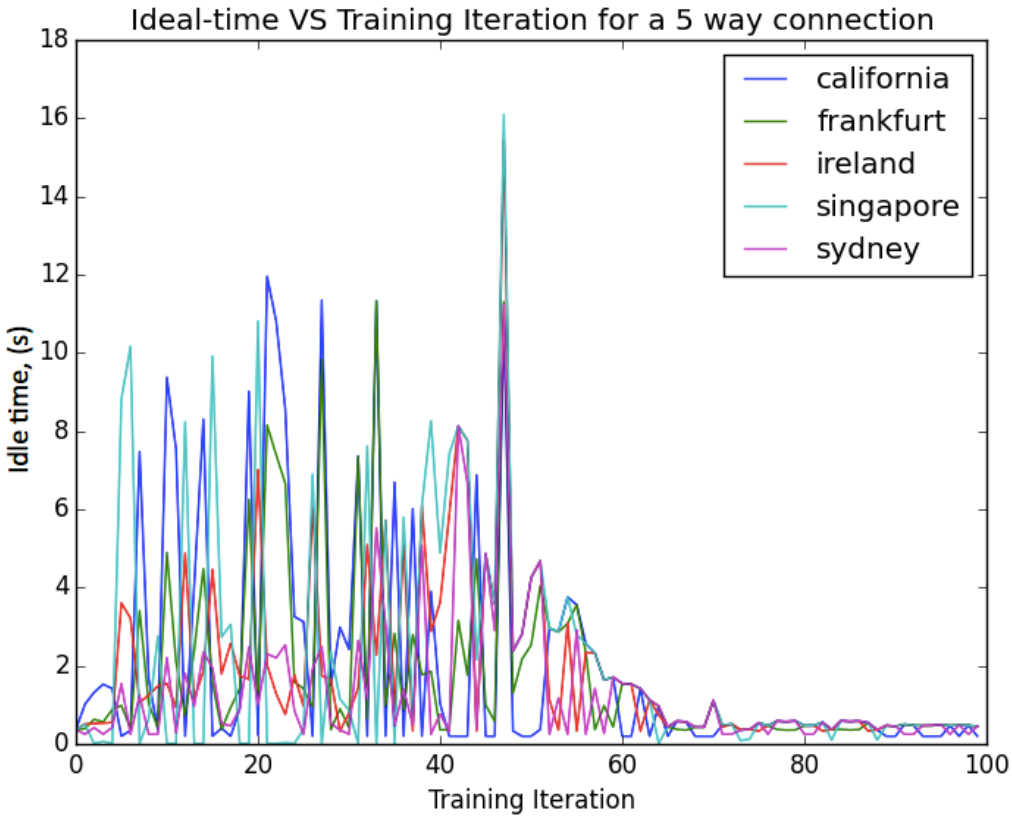


Figure 4.16: The idle time for 5-way multiconnection based on worker server locations.

Table 4.3: The idle time mean and standard deviation (STD) for worker servers located at different cities

Worker Server Location	Idle Time (s) Mean	STD
Sydney	1.339	1.775
Frankfurt	1.759	2.334
Ireland	1.889	2.307
California	2.078	3.117
Singapore	2.161	3.082

As shown in **Figure 4.16** and **Table 4.3**, the idle time for the 5-way connection was approximately 1.8 seconds, making it almost 7 times longer than a one-to-one connection. The recorded idle time also demonstrated that in multiconnection, single network connection speed and stability did not have any significant importance in the overall performance of the worker server that completed executing earlier, as it still had to wait for other worker servers to finish running.

4.6.6 Conclusion for the Cloud-Based SOM

The results obtained in the previous subchapter demonstrate that it is possible to run the SOM training on a global cloud infrastructure. A common SOM model training iteration is usually configured to have a value between 10,000 – 15,000 iterations and **Table 4.3** suggests that the idle time for a single training iteration on the cloud is approximately 1.8 seconds. Then, the total training time for a cloud-based SOM would approximately take 5 – 7 hours in total, which is an acceptable execution time. Thus, training an SOM on a global cloud infrastructure had been proven to be feasible.

CHAPTER 5
SELF-ORGANIZING MAP HYPERPARAMETER

'The real problem is that programmers have spent far too much time worrying about efficiency in the wrong places and at the wrong times; premature optimization is the root of all evil (or at least most of it) in programming.'

(Knuth, 1974, p.671)

5.1 Chapter Introduction

Every machine learning algorithm can be thought of as a modelling analysis, in which the hyperparameter configurations determine the algorithm learning behaviour and output. The self-organizing map (SOM) algorithm has notably more hyperparameters than other clustering algorithms, given that it combines several machine learning tasks into one. Although the algorithm originally developed for unsupervised clustering tasks, it also decomposes (reduces) the feature dimension to a lower dimension (commonly to 2 dimensions) and at the same time, quantizes the instance vector.

To the best knowledge of the author, there is as yet no research that studies the impact of various hyperparameter configurations on the overall SOM model performance. Thus, an established guide on how to configure the SOM hyperparameters does not exist, particularly for a complex and big volume dataset, such as a particle physics dataset. Hence, the objective of this chapter is to determine the set of hyperparameters that can produce the most optimized SOM model for the Higgs and SUSY dataset.

For this reason, a focused study on the SOM hyperparameter; its function and effect towards the modelling outcome should be done accordingly. The findings in this

chapter were used as the guideline for configuring the SOM hyperparameter in the subsequent chapter. Details regarding the developed SOM application are given in the previous chapter. The experiments carried out in this chapter were mainly executed on the UM sifir cluster, described previously in Chapter 3. Meanwhile, information about particle physics is given in Chapter 2. All SOM model hyperparameter configurations shown in this chapter are given in Appendix A.4.

5.2 Developed Hyperparameter

Table 5.1 lists some of the possible SOM hyperparameters and whether or not its effects upon the modelling outcome are studied in this research;

Table 5.1: List of several SOM hyperparameters

Aspect	Hyperparameter	Studied
SOM Map	Map size – centroid/instance ratio	Yes
	Map topology (discrete/continuous)	No
	Map dimension	No
SOM Model Training	Similarity function	Yes
	Learning-rate / Smoothing function (Homogenous/Heterogeneous)	Yes
	Radius-decay function	Yes
	Training length	Yes
Mapping technique	Global cluster creation	No

5.2.1 Centroid and Instance Number

In the beginning of the SOM model development, several aspects regarding the SOM feature-map had to be defined, including the number of centroids, the map topology, and the initial centroid weight-vector value. The ratio between the number of centroids in the SOM map and the number of instances in the dataset is a crucial hyperparameter. In fact, two aspects of the final SOM model would be directly affected by this ratio:

1. A smaller number of centroids will create a model with lower variance, but higher bias
2. The global instance cluster (GIC) boundary between the centroids is less defined for a model with low number of centroids.

To put in a simple context, a higher ratio of centroid number to instance numbers would result in a better SOM model. However, an SOM model with a high centroid number will take significantly more computing resources and execution time. In this research, the standard number of centroids was set at 30 × 30, as depicted in **Figure 5.1 (a)**, as it offered a balance between the resolution and the execution time.

In terms of SOM map topology, this research focused on the square-shaped SOM map, as shown in **Figure 5.1 (a)**, as this kind of map is easier visualize and interpreted. In another research, more complex centroid distribution topology had been demonstrated, for example, most studies conducted by Kohonen employed a hexagonal-shaped distribution, as given in **Figure 5.1 (b)**. The advantage of using a hexagonal topology is that the number of the nearest neighbouring centroids is increased from 4 (square) to 6. An example of continuous SOM topology is the research conducted by Wu and Takatsuka (2005, 2006), in which they used a geodesic topology, as presented in **Figure 5.1(c)**.

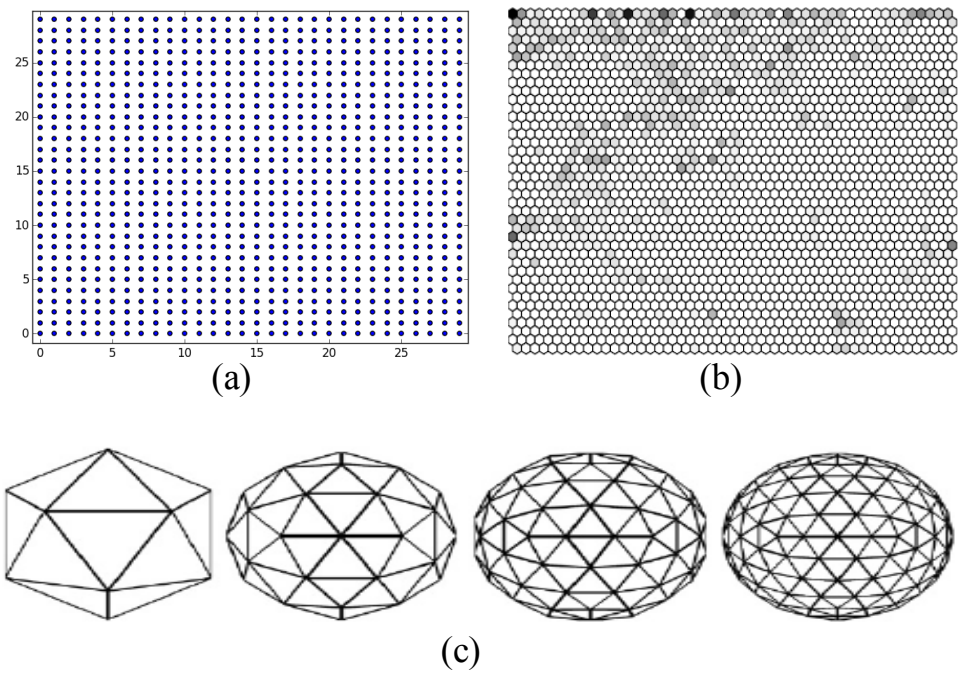


Figure 5.1: Different SOM map topologies from various studies, (a) is the shape of SOM map that had been used in this research, (b) from Kohonen (2013), and (c) from Wu and Takatsuka (2006)

Besides, it is worth noting that by placing the centroid in a hypothetical plane, as in **Figure 5.1**, the SOM centroid acquired a spatial parameter (XY-coordinate) that is independent of the instances value during the training phase.

5.2.2 Training Iteration Length

Configuring the training iteration should take into account the number of instances in the dataset, the number of centroids, and the acceptable run time. Higher training iterations would allow the centroid weight-vector to converge to a value that better reflect the topology of instances in the dataset. A more in-depth study on the effect of training iterations is given in subchapter 5.4.3 In this research, a value between 2,700 and 50,000

iteration steps was taken, depending on how large the dataset and the number of centroids were.

5.2.3 Learning-Rate Function and Radius-Decay Function

The configuration of the learning-rate function has a direct effect on the convergence level of the centroid weight-vector. The convergence level of the centroid weight-vector, in turn, affects how the SOM model clusters instances. Thus, the learning-rate function is a vital hyperparameter for SOM modelling.

To the best knowledge of the author, there have been no studies on the effect of using different forms of learning-rate functions upon the outcome of the SOM model. In this research, four different learning-rate functions had been studied, the logistic regression equation (5.1), reverse logistic regression (5.2), derivative hyperbolic tan (5.3), and a form of a damped sinusoidal wave equation (5.4), where λ is defined in equation (5.5):

$$\frac{1}{1+e^{-\lambda}} \tag{5.1}$$

$$1-\frac{1}{1+e^{-\lambda}} \tag{5.2}$$

$$\frac{1}{(\cosh(-\lambda))^2} \tag{5.3}$$

$$\frac{1}{2}\cdot e^{-kt}\cdot\cos(\omega t) \tag{5.4}$$

$$\lambda = -k(t-\frac{T}{2}) \tag{5.5}$$

Where *T, t*, and *k* are the maximum training length, the training iteration step, and steepness of the curve respectively. The effect of *k* on the equation form is show in **Figure 5.2**. However, the linear function (such as the $kx + C$) was not studied as it was assumed that it would give results comparable to those given by the logistic regression and reverse logistic regression.

In this research, *k* was set to be equal to 10 for logistic and reverse logistic functions, while $k = 5$ for derivative hyperbolic tan and damped sinusoidal wave. These two value was chosen as it produce a curve that is neither too steep (as $k = 100$) or too flat (as $k = 2.5$). For the radius-decay function, this research only used the reverse logistic regression with the minimum value set to 1 centroid.

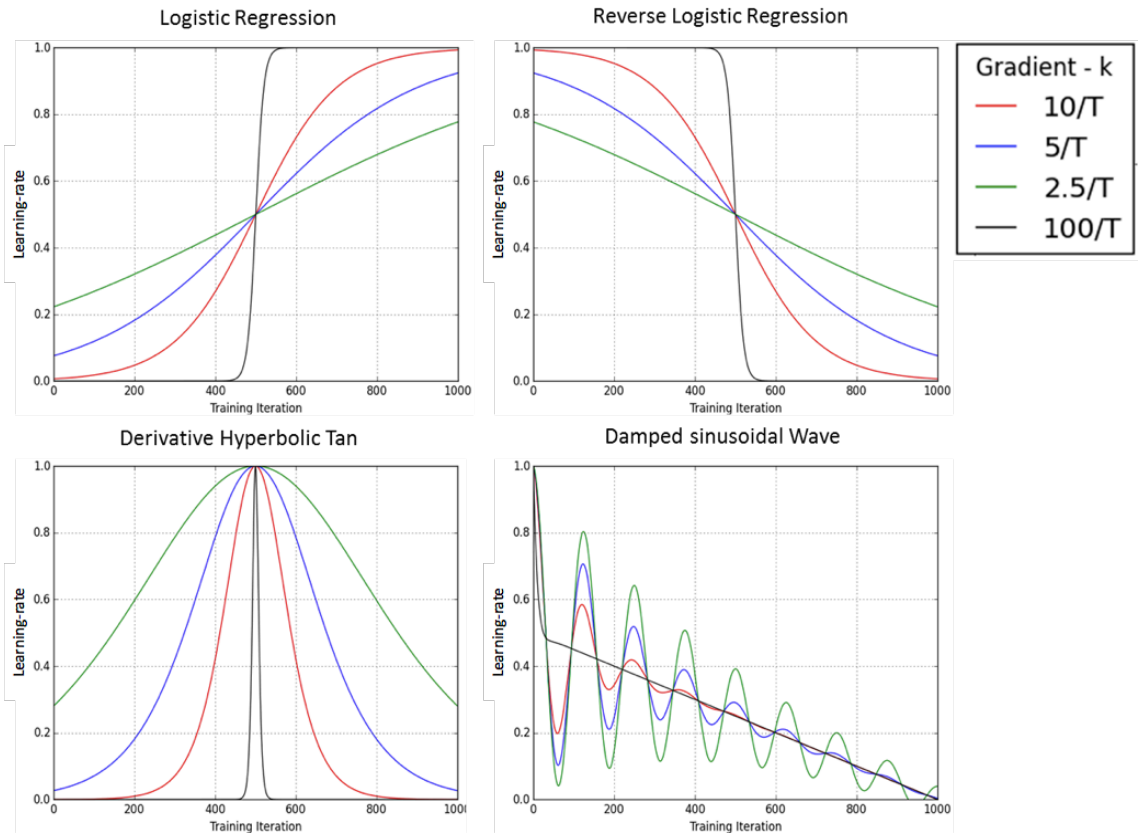


Figure 5.2: Different forms of learning-rate function with different values of *k*.

5.2.4 Homogenous and Heterogeneous Learning-rates

The SOM equation that is shown in equation (4.4) shows that the learning-rate function depends only on the training iteration, thus, the learning-rate value is same (homogenous across the SOM map) for all centroids in each training iteration. However, not all centroid's weight-vector is perturbed in each training iteration. Only centroids with distance less than R_{max} from the winning centroid will have it weight-vector perturbed. Thus, at any given training iteration, different centroids would have different numbers of times its weight-vector have been perturbed.

Figure 5.3 shows the ratio between the number of times a centroid was perturbed in the training phase and the maximum training iteration, across the SOM map. It is apparent that the ratio was not homogenous across the map, in which centroids closer to the edges were perturbed less than those at the centre.

This condition should be take into account in creating the SOM model by changing the factor on which the learning-rate function from the training iteration, t , into the number of times a centroid has been perturbed in the previous training iteration, t_s, via equation (5.6). In this way, the learning-rate values would be specific to each centroid and the learning-rate value is heterogeneous across the SOM feature-map.

$$l(t) \rightarrow l(t_s) \tag{5.6}$$

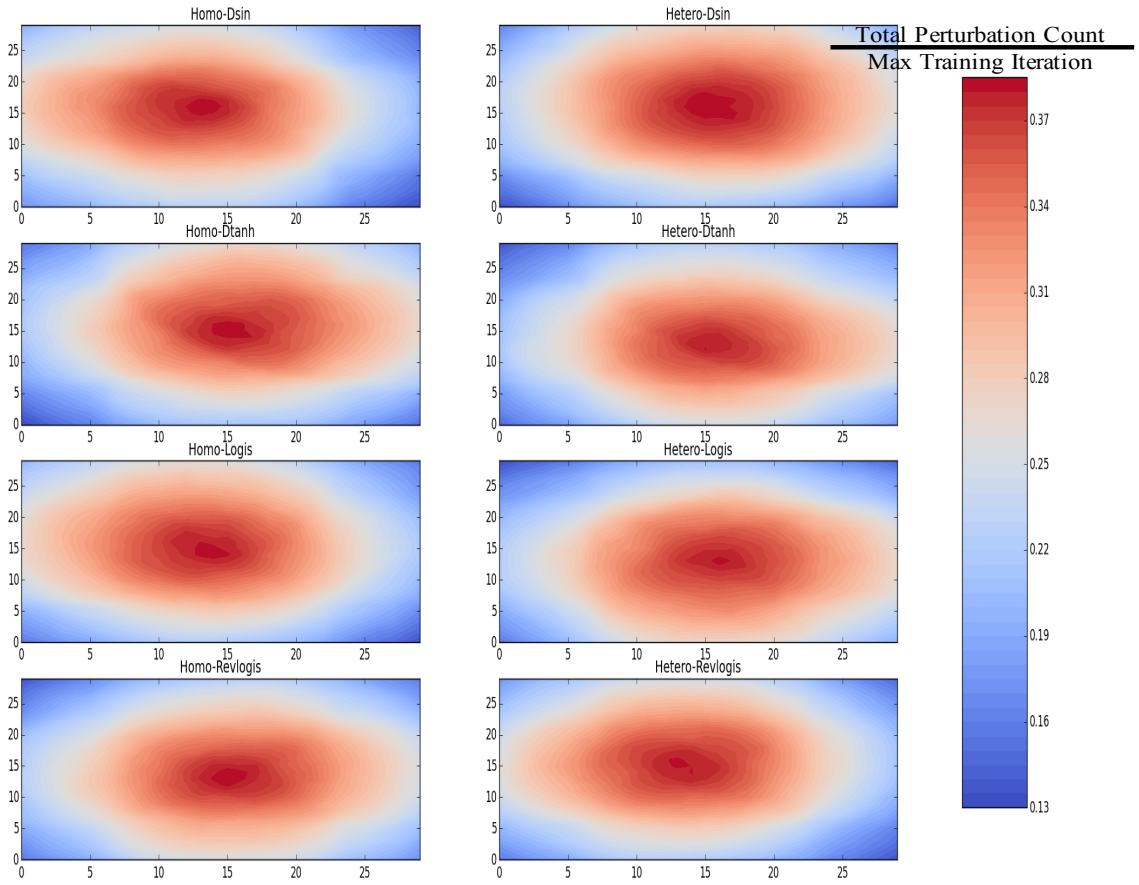


Figure 5.3: The ratio between the number of times a centroid had been perturbed in the training phase to the maximum training iteration

For example, a SOM model was being created with its training iteration is set to be 1000 steps. An arbitrary centroid located in the middle of the SOM map is label as 'A' while another arbitrary centroid located at the edge of the SOM map is label as 'B'. During the 500th training iteration, centroid-A weight-vector has already been perturbed 480 times while centroid-B weight-vector has only been perturbed only 200 times. If a homogenous type learning-rate was used, then both centroid-A and centroid-B will have their learning-rate equal to $l(500)$ in their 500th training iteration. On the other hand, for a heterogeneous type learning-rate mode, centroid-A will have a learning-rate equal to $l(480)$ while centroid-B will have learning-rate equal to $l(200)$.

In **Figure 5.3**, changing the learning-rate from a homogenous (Homo, left column) to a heterogeneous (Hetero, Right column) learning-rate function did not change the overall distribution shape. In the figure, the derivative hyperbolic tan had been denoted as *Dtanh*, reverse logistic regression as *Revlogis*, damped sinusoidal wave as *Dsin*, and logistic regression as *Logis*, equation (5.1-5.5).

However, **Figure 5.4** shows that there is an apparent difference in the total learning-rate for by each centroid between homogenous and heterogeneous learning-rate modes. For derivative hyperbolic tan (Dtanh) and logistic regression (Logis) learning-rate functions, the learning-rates were more homogenous (a bigger red spot) when they were in a homogenous mode. Meanwhile, damped sinusoidal wave (Dsin) and reverse logistic regression (Revlogis) learning-rates were more homogenous when they were used in a heterogeneous mode.

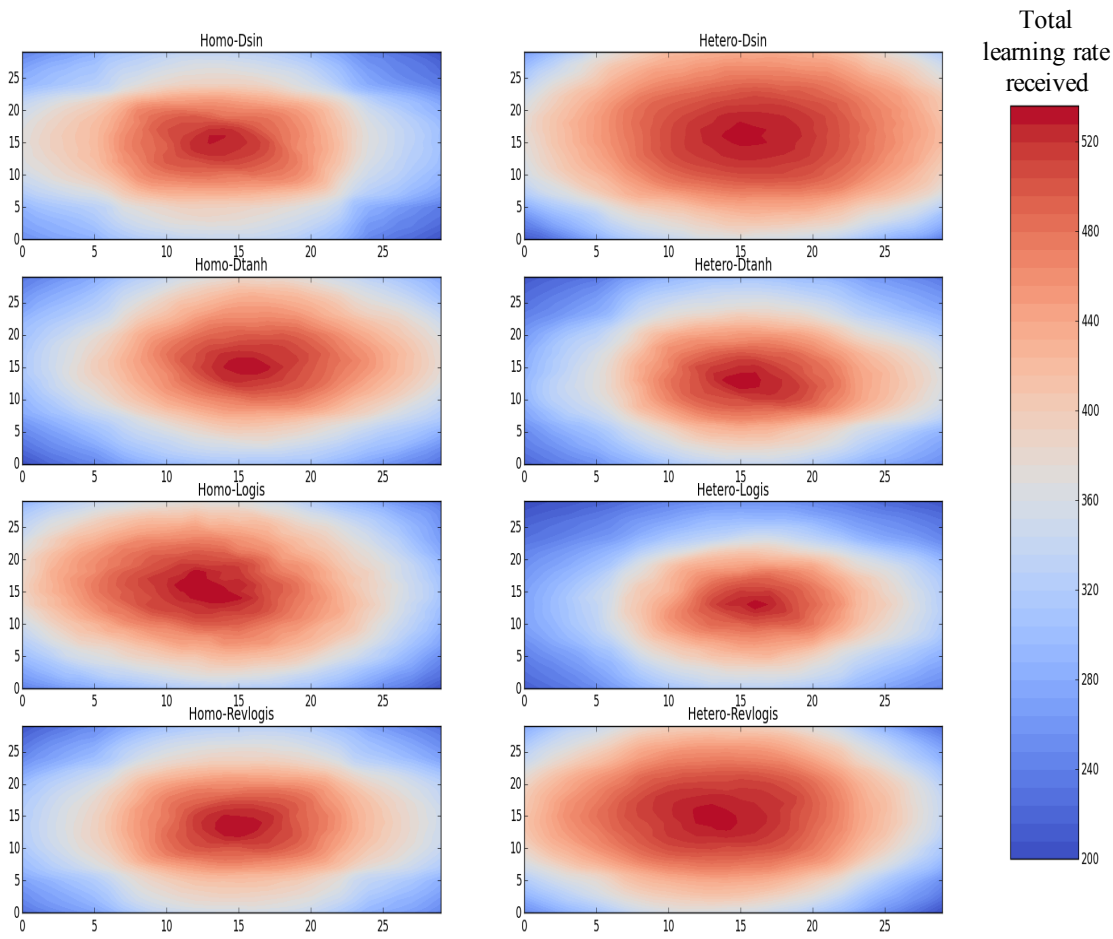


Figure 5.4: The total learning-rate distribution for each learning-rate function in both homogenous (Homo) and heterogeneous (Hetero) modes

5.2.5 Similarity Function

A distance function in a certain feature space can be used to measure the similarity between two vectors. In most ML literature, the terms 'distance function' and 'similarity function' are used interchangeably. The SOM algorithm uses the similarity function to select the winning-centroid in each training iteration.

This research studied the effect of using various similarity/distance functions on the SOM modelling outcome. Previous studies by other authors have studied the impact

of using Euclidean distance, dot product, and Cosine similarity on SOM modelling. This work expands this to include other distance functions, as shown in **Table 5.2**.

A summary of these functions is given subsequently in the text, nevertheless, readers who are interested in an in-depth discussion are directed towards publications made by Cha (2007), Wang and Sun (2015), as well as Yang and Jin (2006). These publications surveyed the different types of distance/similarity functions and their application to machine learning and data mining.

It is vital to point out here that in using a similarity function, the smaller the value obtained, the more similar the two vectors are, whereas the higher the value obtained, the more dissimilar the two vectors are.

Firstly, the similarity functions that fall under the Minkowski family are given in equation (5.7) (Cha, 2007):

$$\text{distance} = \sqrt[p]{\sum_{i=1}^{d} |u_i - v_i|^p} \tag{5.7}$$

The Minkowski distance is the generalization of the city-block (Manhattan), Euclidean, and Chebyshev distances, where each corresponds to a p value of 1, 2, and approaching ∞ respectively. Kohonen (2013) stated that Euclidean distance is the standard choice for constructing the similarity matrix in SOM modelling.

Table 5.2: Similarity/distance functions studied in this research, all functions were taken from Cha (2007), except correlation distance, which had been taken from the scipy module

Formula Name	Formula
Euclidean Distance	$d = \sqrt{\sum_{i=1}^{d} \lvert u_i - v_i \rvert^2}$
City-block Distance	$d = \sum_{i=1} \lvert u_i - v_i \rvert$
Correlation Distance	$1 - \frac{(u - \bar{u}) \cdot (v - \bar{v})}{\lVert (u - \bar{u}) \rVert \lVert (v - \bar{v}) \rVert}$
Cosine Distance	$d = \frac{\sum_{i=1}^{d} u_i v_i}{\sqrt{\sum_{i=1}^{d} u_i^2} \sqrt{\sum_{i=1}^{d} v_i^2}}$
Chebyshev Distance	$max_i \lvert u_i - v_i \rvert$

As for the city-block function, the equation behaves well for instances with high dimension, since city-block function has a p value of 1, increasing the similarity measurement (generally) as the dimension increases. This property has the effect of amplifying minute differences between the two instances, even though such differences are scattered across multiple features. As such, the city-block measures the differences between the instances in terms of their magnitude rather than their direction.

As for the cosine similarity function, it is frequently used by text processing software, where the count of each word in a given text can be used to determine the genre the text belongs to (Wang & Sun, 2015). The function measures the angle between the two vectors rather than its distance, thus providing a means to gauge the difference in the

direction between the two instances. This method is very useful if the variation in a dataset feature is high and separable only by using the instance direction in the feature space.

The last distance formula presented is the correlation distance, which measures the independence of one instance from another. The function is not a pair-wise distance measurement as cosine and Minkowski functions are, but a similarity measurement over a sample of instances. Thus, it is more closely related to functions, such as Mahalanobis, Sørensen, and Canberra distances. In principle, it is a misuse to construct an SOM distance matrix with the correlation distance formula, however, the author was interested to see how it affected the SOM module outcome.

5.3 Higgs Dataset Feature Engineering

Each similarity function can only measure the similarity among instances for finite number of dimensions. As the feature dimension increases, the measurable dissimilarity between two vectors will diminish, a phenomenon commonly known as the 'curse of dimensionality' (Bellman, 1961). Thus, in machine learning, feature engineering is done prior to ML modelling. In feature engineering, the features of a dataset are cherry-picked, as well as transformed and/or decomposed to reduce its dimension. In certain cases, the results from feature engineering dictate the machine learning algorithm that is suitable for the dataset.

The objectives of the following subchapter are:

1. To identify the similarity function that can provide optimum dissimilarity between signal and noise instances at high feature dimension for the Higgs dataset

2. To identify the feature group that provides optimum dissimilarity between signal and noise instances for the Higgs dataset

Only the Higgs dataset had its features analysed because it was the only dataset that was used in the supervised learning method, hence, the label of each instance was identified by the learning algorithm. This also allowed the differences in similarity distribution between signal and noise instances to be measured before the model training.

The Higgs dataset feature can be grouped into several groups, as shown in **Table 5.3** (Baldi et al., 2014).

Table 5.3: Higgs dataset feature groups

High-level features	Raw features					
	Lepton	Missing energy (MET)	Jet 1	Jet 2	Jet 3	Jet 4
m_{jj}	lepton pt	missing energy magnitude	jet1 pt	jet2 pt	jet3 pt	jet4 pt
m_{jjj}	lepton eta	missing energy phi	jet1 eta	jet2 eta	jet3 eta	jet4 eta
m_{lv}	lepton phi		jet1 phi	jet2 phi	jet3 phi	jet4 phi
m_{jlv}			jet1 b-tag	jet2 b-tag	jet3 b-tag	jet4 b-tag
m_{bb}						
m_{wbb}						
m_{wwbb}						

Group	Feature Group
Group 1	High-level
Group 2	High-level + Lepton
Group 3	High-level + Lepton + MET
Group 4	High-level + Lepton + MET + Jet1
Group 5	High-level + Lepton + MET + Jet1 + Jet2
Group 6	High-level + Lepton + MET + Jet1 + Jet2 + Jet3
Group 7	All

The SUSY and the Dimuon datasets were used in the unsupervised learning method. In this method, the label for each instance was removed during the SOM model training. Without the label, the features or the feature groups that provided the best

separation between signal and noise could not be established, therefore, feature engineering could not be conducted in this situation.

5.3.1 Measurement Method for the Single Higgs Sub-dataset

This subchapter describes study on how the increase in the feature dimension affected the similarity measurement between the signal instance and the noise instance from the Higgs dataset.

First, the study limits its measurement to only three Higgs sub-datasets. Each dataset contained only 300 randomly chosen instances. Two of the sub-datasets contained only noise instances, and were labelled as 'ctr-noise', and 'test-noise', while the third was created only for signal instances and was labelled as 'test-signal'.

Initially, each sub-dataset feature was limited to the high-level feature group shown in Table 5.3. After preparing these three sub-datasets, the following steps were taken:

1. An instance from the ctr-noise was chosen and the this similarity of this instance to all instances in the test-noise was measured using the Euclidean distance function. The obtained values were recorded.
2. The previous step was repeated all other instances in the ctr-noise.
3. Steps 1 and 2 were repeated with the test-noise replaced with the test-signal sub-dataset.
4. Steps 1, 2, and 3 were repeated with the feature dimension increased for all sub-datasets by including another feature group in **Table 5.3**. This step was repeated until all feature groups were included.

5. Steps 1, 2, 3, and 4 were repeated with the Euclidean distance replaced with other similarity function.

5.3.2 Measurement Results for The Single Higgs Sub-dataset

The recorded similarity value for a given similarity function is shown as a cumulative distribution in **Figure 5.5** - **Figure 5.9**. In each figure, the subplots (a) - (g) represent the cumulative combination of groups 1 - 7 respectively, as listed in **Table 5.3**. In addition, the absolute mean of the difference between noise-noise and noise-signal similarity measurements versus the feature dimension number is shown in subplot (h). Note that any result that comes from this measurement is specific to the three Higgs sub-datasets only.

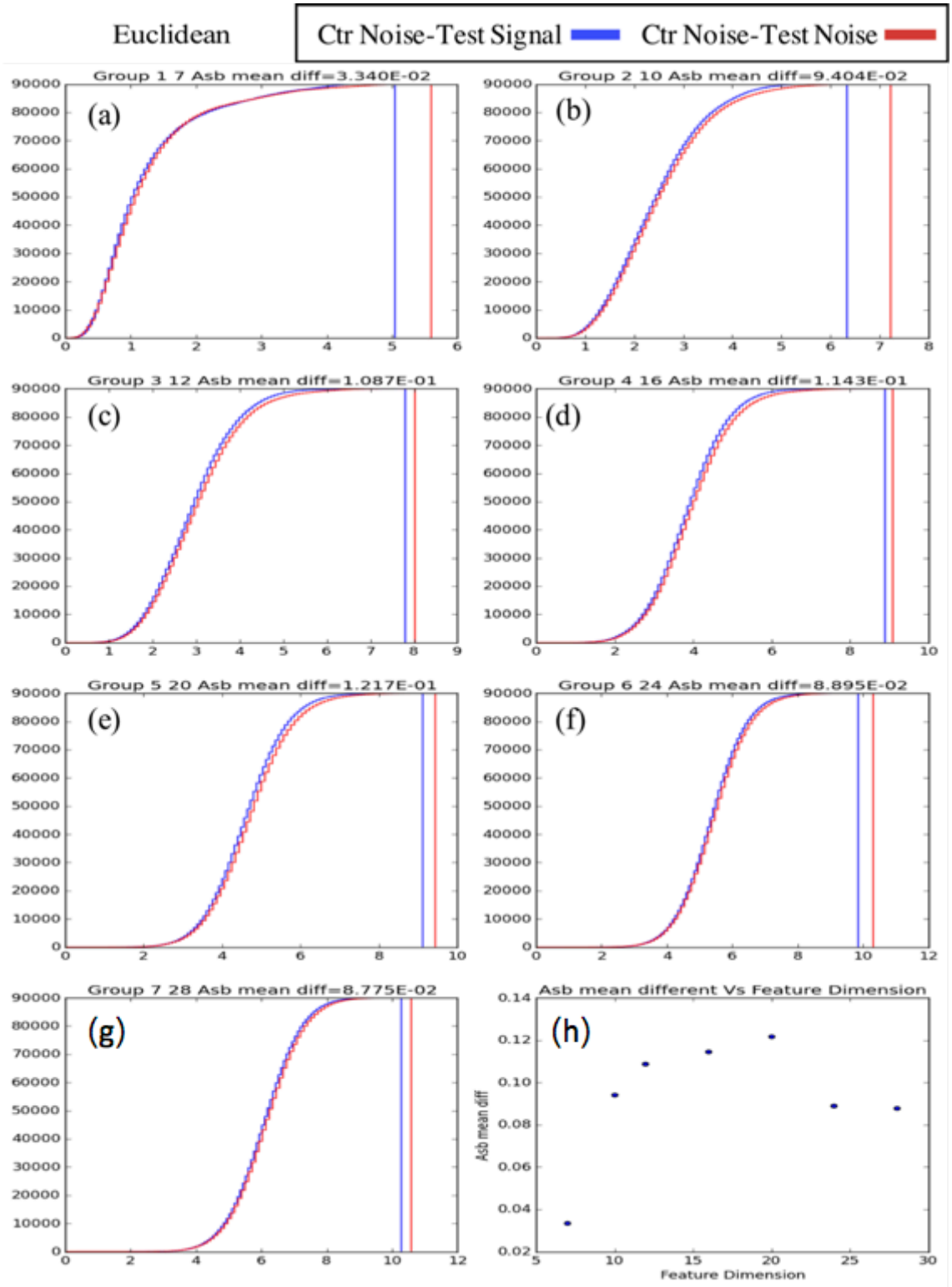


Figure 5.5: The cumulative distribution of similarity between noise-noise and noise-signal under different feature combinations for Euclidean Distance (a-g). (h) the scattered plot of absolute average different versus the feature dimension.

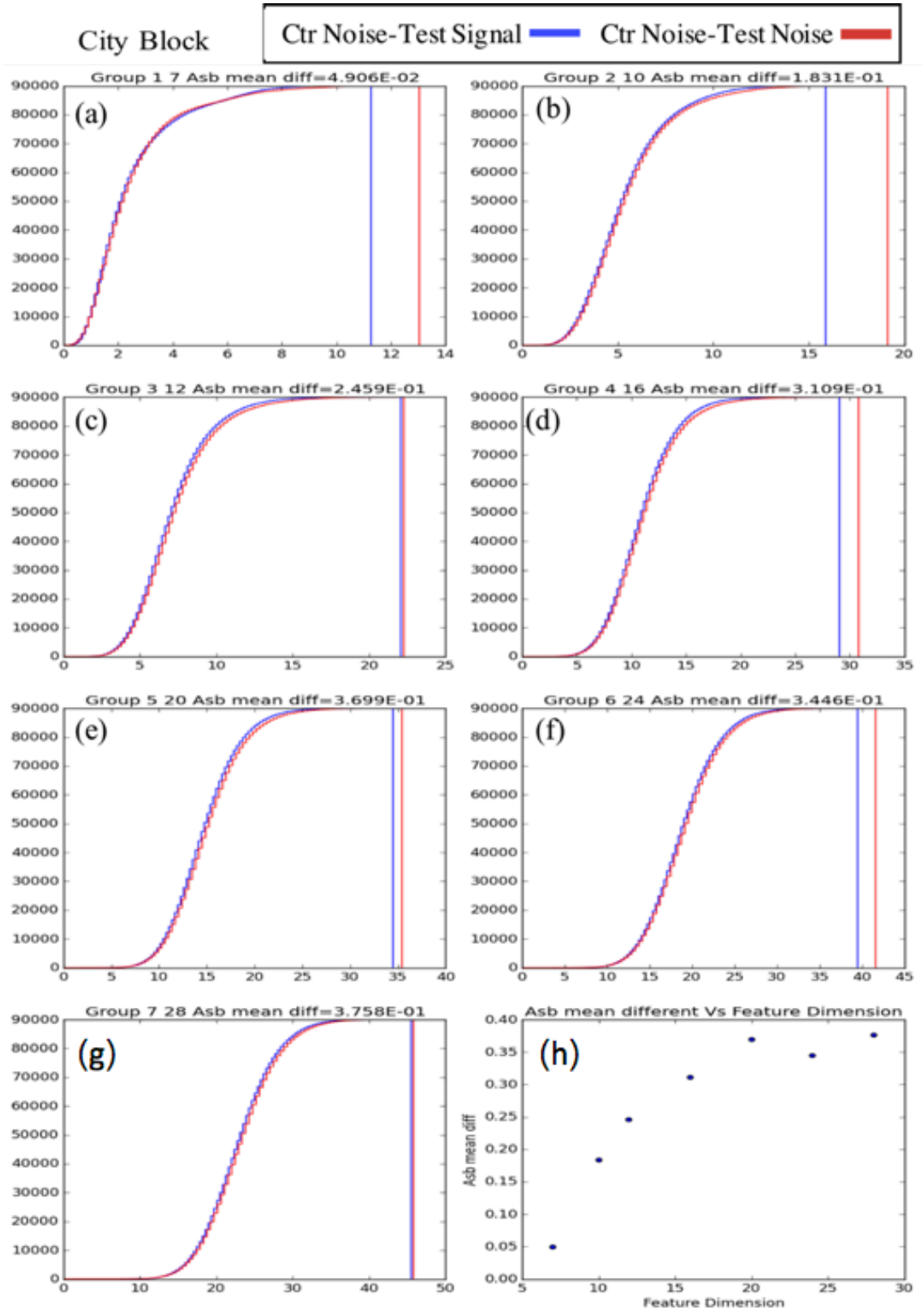


Figure 5.6: The similarity measurement cumulative distribution between noise-noise and noise-signal under different feature combinations for City-block (a-g). (h) the scattered plot of absolute average different versus the feature dimension.

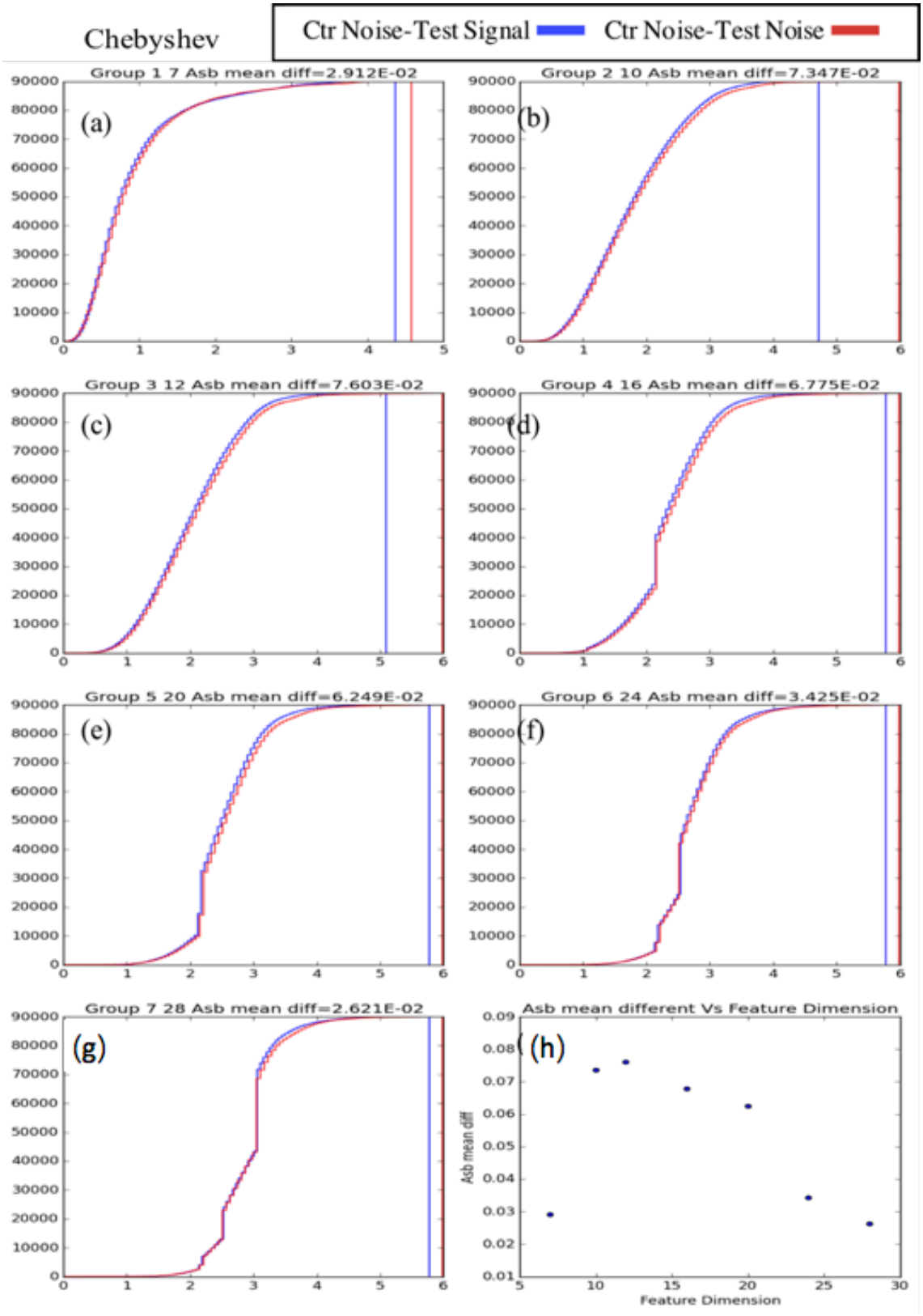


Figure 5.7: The similarity measurement cumulative distribution between noise-noise and noise-signal under different feature combinations for Chebyshev (a-g). (h) the scattered plot of absolute average different versus the feature dimension.

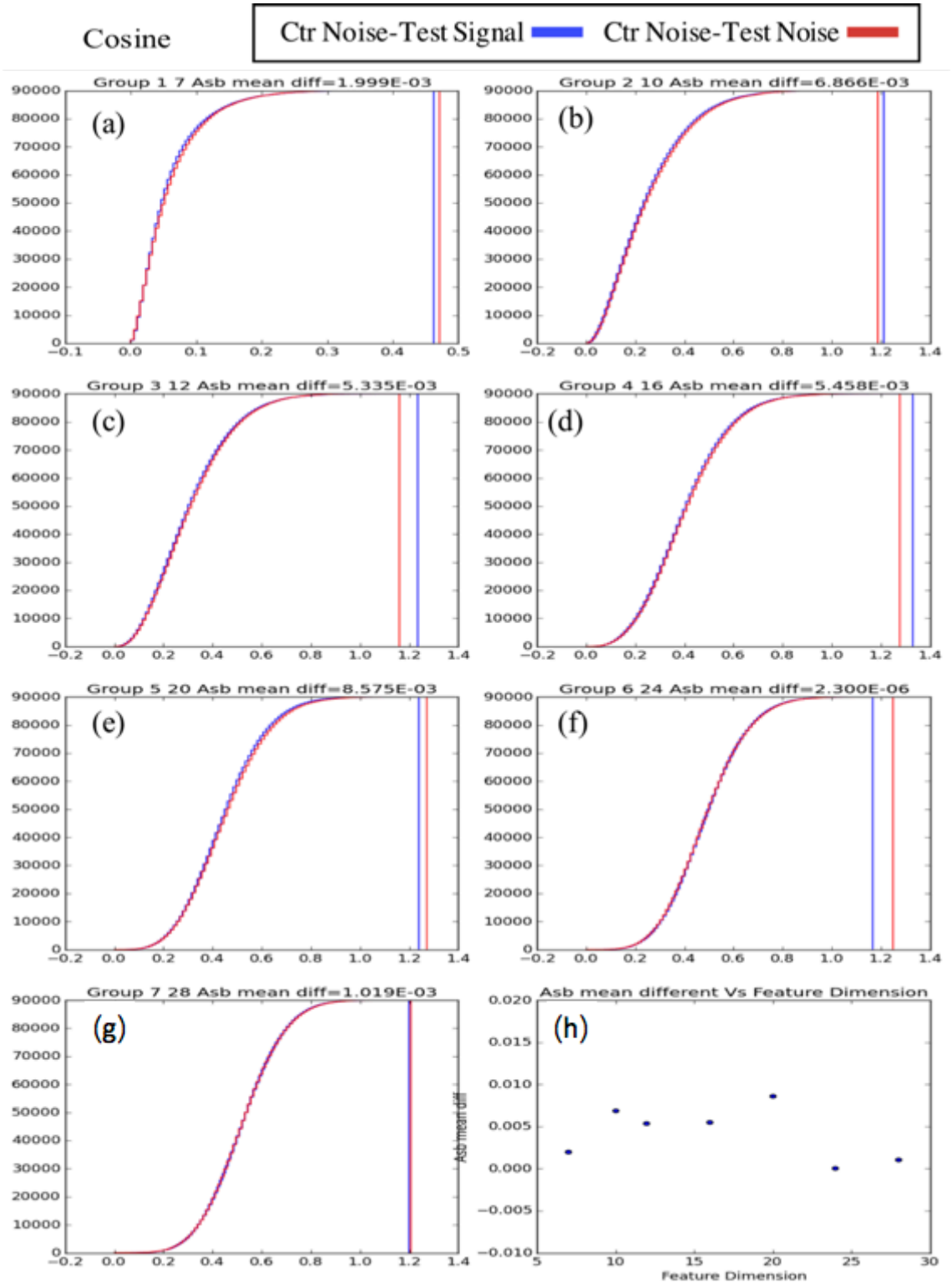


Figure 5.8: The similarity measurement cumulative distribution between noise-noise and noise-signal under different feature combinations for Cosine (a-g). (h) the scattered plot of absolute average different versus the feature dimension.

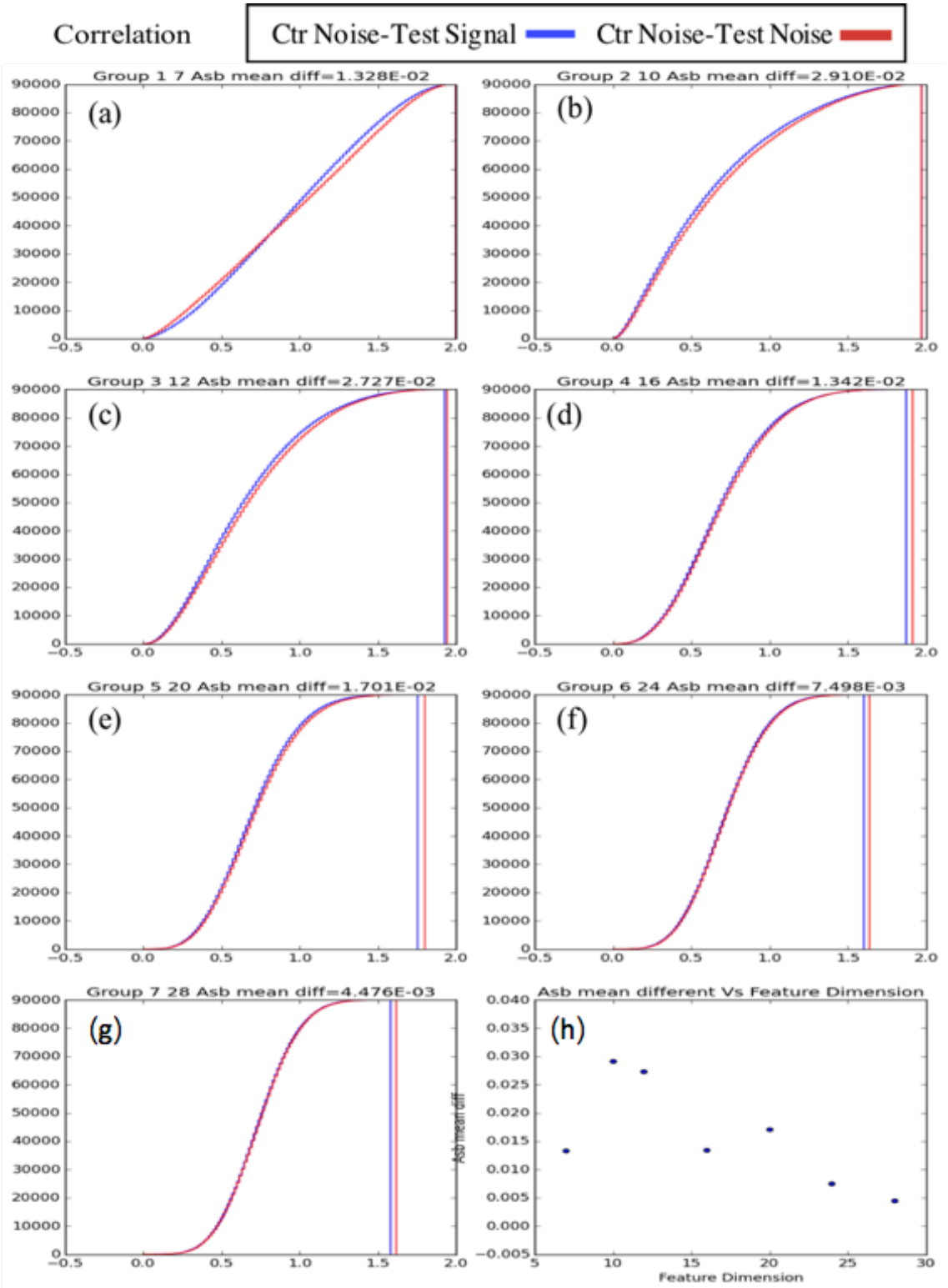


Figure 5.9: The similarity measurement cumulative distribution between noise-noise and noise-signal under different feature combinations for Correlation (a-g). (h) the scattered plot of absolute average different versus the feature dimension.

5.3.3 Discussion for Higgs Sub-Dataset Feature Engineering

An idle similarity function will give a value that approaches zero between two noise instances, but gives a high positive value between a signal and noise instance. The result for each similarity function is discussed in the following:

Euclidean Distance: The cumulative plot shown from **Figure 5.5 a-h**, shows that the high-level features (listed in **Table 5.3)** did not give a high measurable dissimilarity between noise and signal instances. Only a small portion of signal instances had a higher magnitude in certain feature(s) when compared to noise instances, which contributed to the slight increase of the noise-signal curve (blue) after similarity magnitude of 4. For Euclidean distance, the Lepton group give the highest dissimilarity, follow by (in no particular order) MET, Jet 1, and Jet 2. On the other hand, Jet 3 group dramatically decreased the dissimilarity, and Jet 4 group appeared do not contribute to any significant effect in **Figure 5.5(h).**

City-block Distance: The cumulative distribution of **Figures 5.6 b-e,** show that the noise-signal similarity curve (blue) was regularly above the noise-noise similarity curve (red), indicating that the majority of the noise-signal similarity measurement had a lower magnitude than the noise-noise measurement. **Figure 5.6 (h)** also shows that the absolute mean difference between this distribution increased as more feature groups were added until it plateaud at a dimension > 20. Similar to the Euclidean distance measurement, the high-level features did not give any significant dissimilarity between noise and signal instances, whereas the Jet 3 group decreased the dissimilarity.

Chebyshev Distance: The cumulative distribution of similarity measurement by using the Chebyshev function was similar to the output produced by the Euclidean and City-block, as shown in **Figure 5.7 (a)**. The highest dissimilarity increased between the noise and the signal due to the contribution of the Lepton group and peaked when the MET group was incorporated. Incorporating additional dimensions only decreased the dissimilarity measurement, thus, it can be said that Chebyshev distance was insensitive to dissimilarity (for this sub-dataset) at high dimension. The discontinuities in **Figure 5.7 d – f** was due to the b-tag parameters which have discrete values, with a maximum value of 2.17307.

Cosine Similarity: The cosine similarity measures the angle difference between two vectors. The cumulative similarity distribution of **Figures 5.8 a – f** have similar characteristics as the previous two cumulative similarity distribution figures. The only difference is that the absolute mean difference scatter plot, **Figure 5.8 (h)**, shows that angle difference had been the highest when the Lepton and the Jet 2 groups were incorporated into the feature, whereas other feature groups exhibited a weaker or declining effect. However, the increase in feature dimension did not affect greatly the dissimilarity measured by using these functions.

Correlation Distance: The cumulative plot of **Figure 5.9 (a)** shows that a certain portion of the noise-signal curve was below the noise-noise sub-dataset. This indicated that by using the only high-level feature with correlation distance function, certain noise instances were measured to be more similar to a signal instance than another noise instance, making the result obtained to have a higher error level than any other similarity function. Moreover, **Figure 5.9 (h)** shows that the function had higher sensitivity towards increase in dimension than any other similarity functions. In spite of this, the function

still showed a high jump in dissimilarity value for the Jet 3 feature group and a steep increase for Lepton feature group.

The results obtained here show that different combinations of features and similarity functions led to different similarity values between the noise and the signal instances. This is specifically for the three sub-datasets, the Lepton and the Jet 2 feature groups displayed the highest dissimilarity measurement between signal and noise instances. Moreover, the Cosine similarity, the Euclidean distance, and the City-block distance functions showed less sensitivity to an increase in feature dimension, whereas Chebyshev and Correlation performances dropped significantly as the dimension increased.

5.3.4 Multiple Higgs Sub-Dataset Similarity Measurement

In the previous subchapter, the measurement had been limited to three sub-datasets. In this subchapter, the measurement is generalized. Multiple new sub-datasets were created for each type of sub-dataset (ctr-noise, test-noise, and test-signal). The same method described in subchapter 5.3.2 was then repeated for all new sub-datasets.

The mean for the absolute difference between noise-noise similarity values and noise-signal similarity values for each of these datasets are shown in the scatter plots in **Figure 5.10** and **Figure 5.11**.

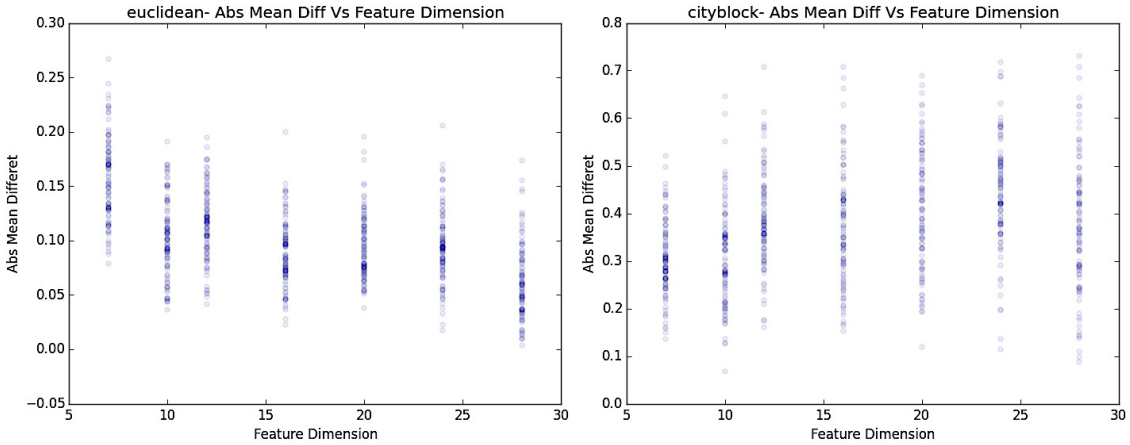


Figure 5.10: The absolute difference in mean between noise-noise similarity values and noise-signal similarity versus feature dimension number for Euclidean and City-block functions

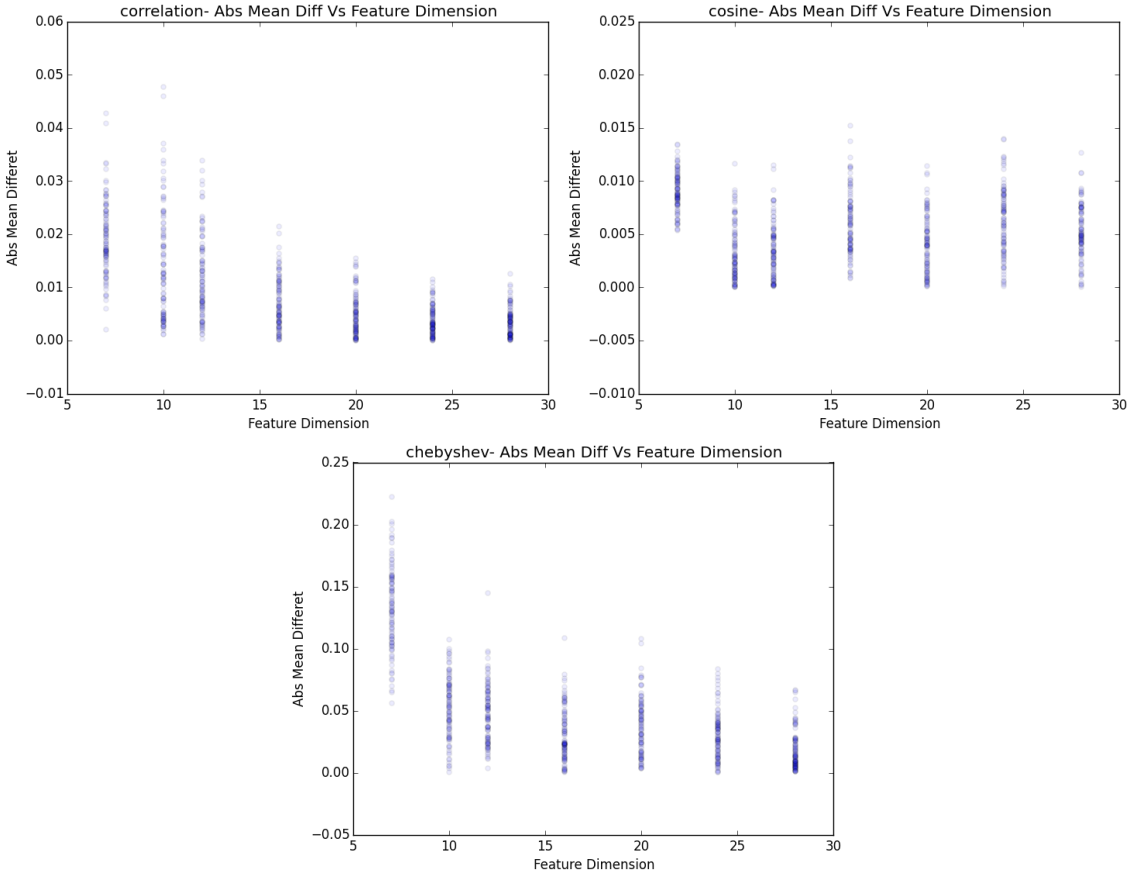


Figure 5.11: The absolute difference in mean between noise-noise similarity values and noise-signal similarity versus feature dimension number for Correlation, Cosine, and Chebyshev Functions

The results shown in **Figure 5.10** and **Figure 5.11** did not indicate that the Lepton and the Jet feature groups (points at dimension = 10 & 20 respectively) offered better discrimination than other feature group combinations. Furthermore, the higher-level feature group (**Table 5.3**, corresponding to points at dimension = 7) showed a stronger dissimilarity measurement for the Euclidean and Chebyshev than expected from the previous result.

The primary objective of this subchapter is to demonstrate that for the Higgs Dataset, no one feature combination can consistently provide the largest dissimilarity measurement between signal and noise instances. Each smaller dataset (sub-dataset) instances had individual feature combinations that gave maximum dissimilarity.

In the context of machine learning practice, this is a strong indication that the Higgs dataset required algorithms that leveraged information available at a subspace level, please see Skurichina & Duin, (2002,) for discussion on subspace classifier. The best machine learning methods that can do this are 'Ensemble' type algorithms, such as the Random Forest algorithm, which use the bagging method to create subspaces. The SOM algorithm, indirectly, creates subspace clustering by diversifying the centroid weight-vector during the training phase.

In addition to these, **Figure 5.10** and **Figure 5.11** shows that the Euclidean, the city-block, and the cosine was less affected by the increase in dimension, compared to the Chebyshev and Correlation functions. Thus, only these three functions (Euclidean, city-block, and cosine) should be used in the Higgs dataset for SOM modelling.

5.4 Subspace in SOM Feature-map

Subspace clustering is the ability for an ML algorithm to change dynamically its feature(s) selection for a given instance that it is trying to cluster or classified. The objective of this subchapter is to demonstrate the subspace property in an SOM map.

The SOM training phase transforms the initially random weight-vector in the SOM map, **Figure 4.4 (Right)**, into a map with weight-vectors arranged in a smooth manner. **Figure 5.12** shows the magnitude of each centroid weight-vector on the Higgs SOM map that was trained with different learning-rate functions. The distribution of certain features on the trained SOM map is shown in **Figure 5.13** for the Higgs dataset, while **Figure 5.14** shows the distribution of the trained SUSY SOM model.

Each plot in **Figure 5.13** and **Figure 5.14** shows that the distribution of the weight-vector's magnitude is complex and non-linear. The magnitude of a particular feature was high in some regions of the SOM map, while small in others. A large magnitude indicates that a particular feature has a significant influence in clustering instance. Conversely, a low magnitude shows that the feature is redundant for clustering instances in that area of the SOM map. Thus, the feature(s) used for clustering instances can dynamically change across the SOM map.

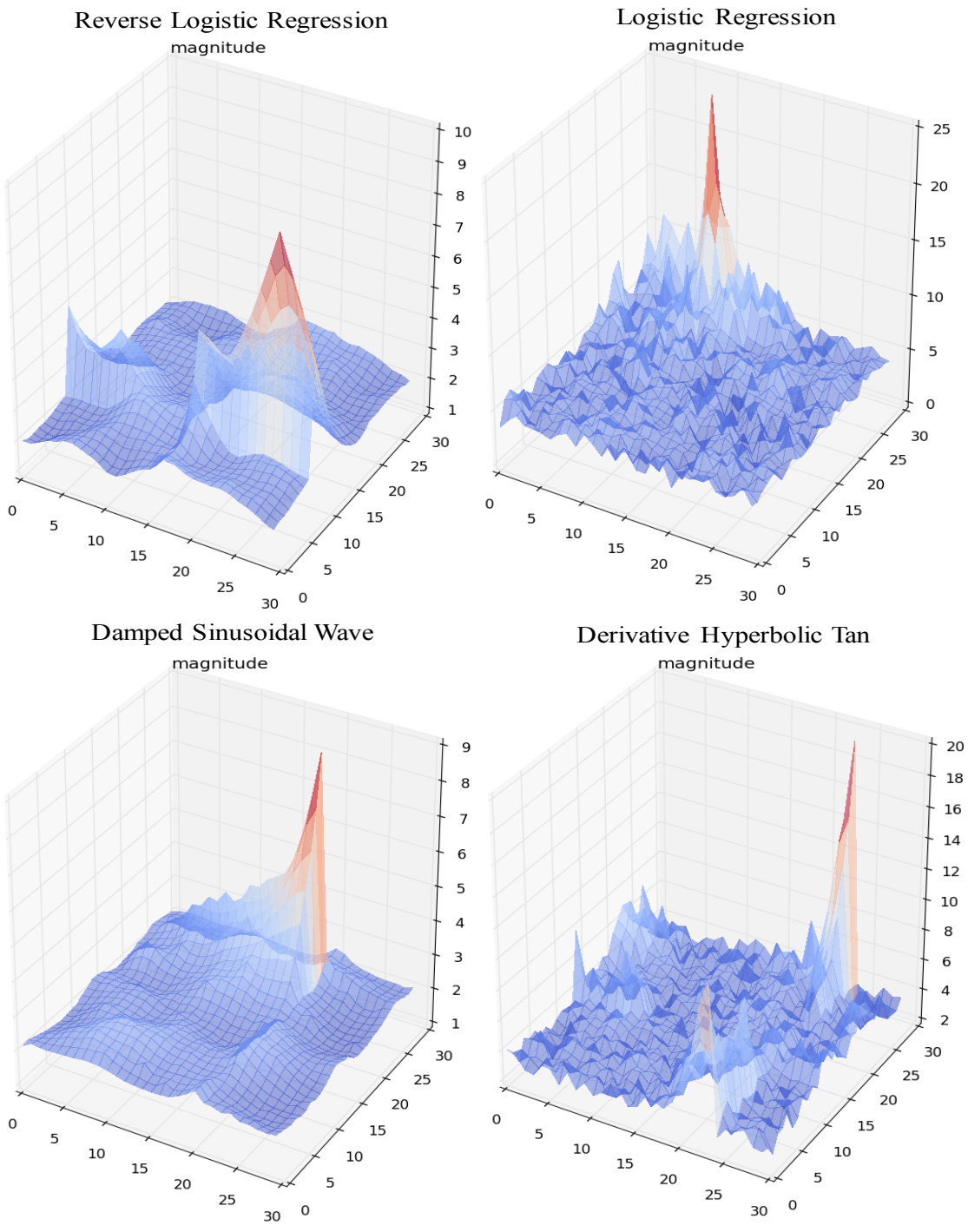


Figure 5.12: The distribution of centroids vector magnitude for different SOM maps, trained with different learning-functions. X-axis and Y-axis refer to the position of each centroid on the XY plane of the SOM map. The Z-axis and colour share the same scale, which indicate the weight-vector magnitude corresponding to a particular point on the SOM map.

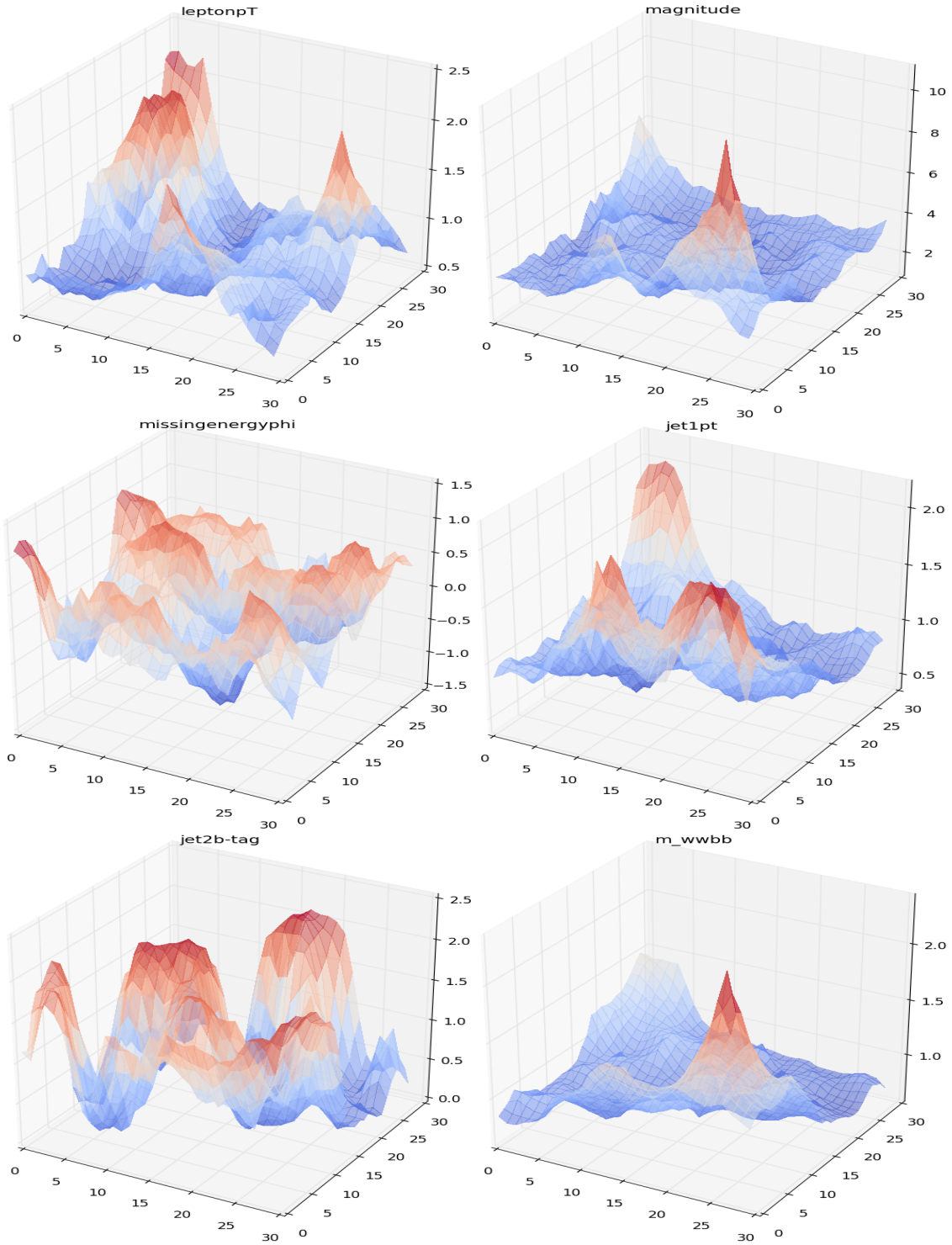


Figure 5.13: The distribution of various features on the trained SOM map for the Higgs dataset. Please see **Figure 5.12** for information about the scale.

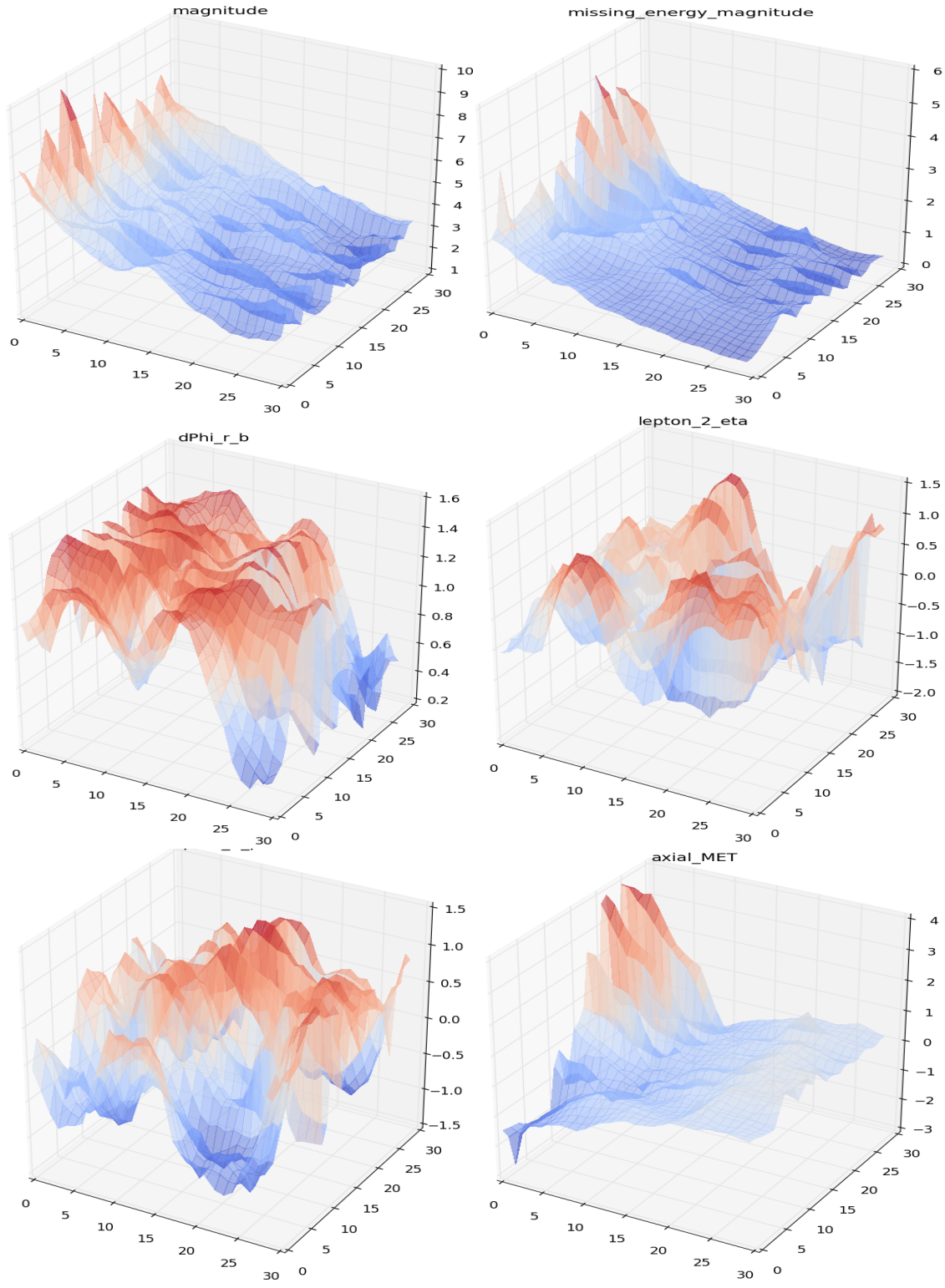


Figure 5.14: The distribution of various features on the trained SOM map for the SUSY dataset. Please see **Figure 5.12** for information about the scale.

For clarity, the following example is given, **Figure 5.15** shows the lepton pt, MET, jet 1 pt, and jet 2 pt feature-maps that belong to the same Higgs SOM model. Two regions, compromising centroids along the x = 3 and x = 23 axes, are marked by the black solid and dashed line respectively. The combination of feature value for each of these centroids, along with these two axes, is shown in **Figure 5.16.**

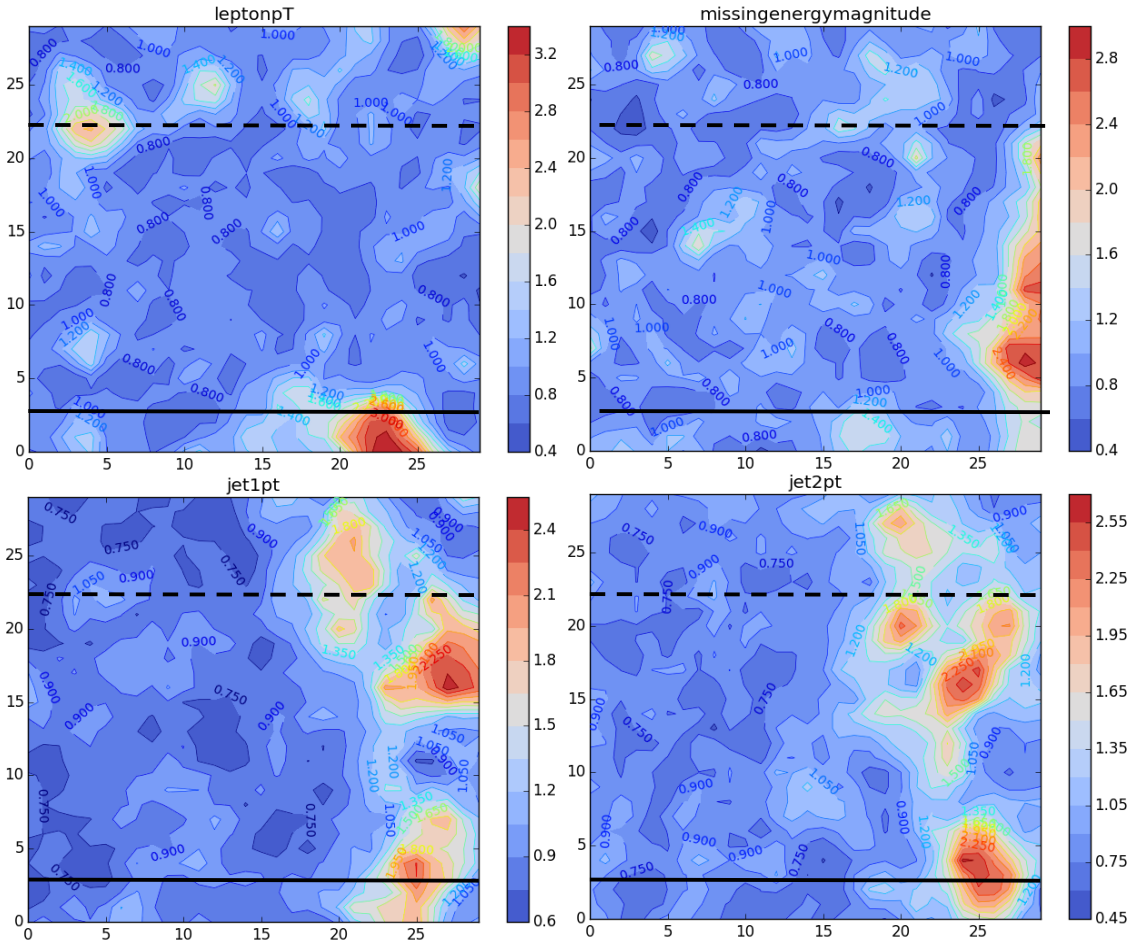


Figure 5.15: The feature-map for lepton PT, MET, Jet1 PT, and Jet2 Pt of the Higgs SOM. Values for each of the features for **Figure 5.16** is taken along the dashed (x = 23) axes and solid line (x = 3).

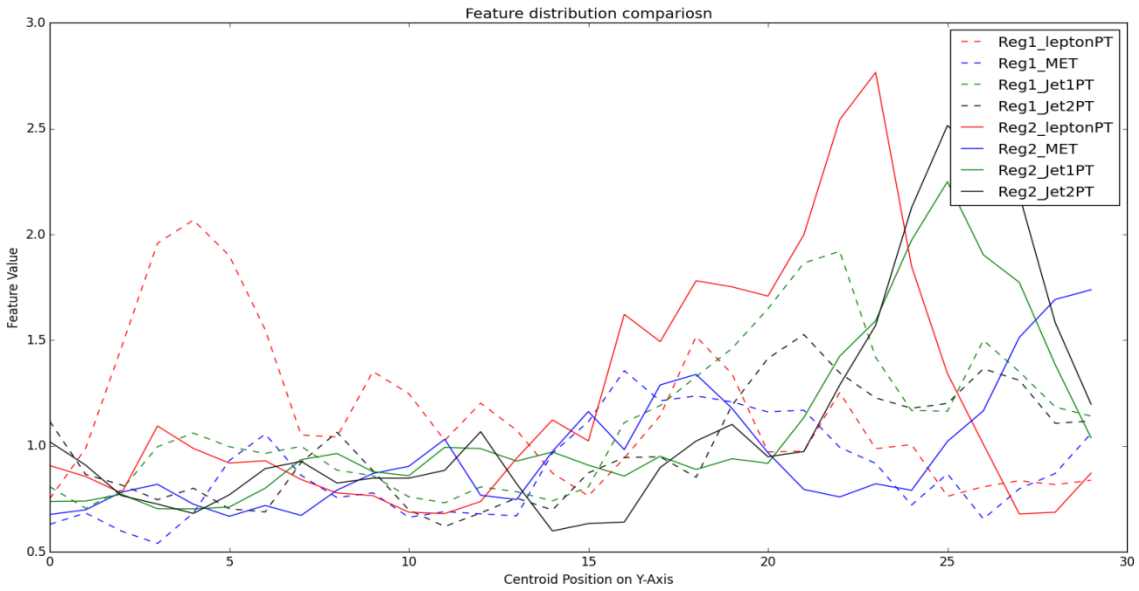


Figure 5.16: The value of 4 different features for each centroid along the x = 3 (Reg1) and x = 23 (Reg2) axes from the SOM model in **Figure 5.15**.

As shown in **Figure 5.16**, each centroid along the x = 3 and x =32 axes had a different combination of values for each feature. Instances with a higher value of Lepton Pt than any other feature value will be mapped in region $0 < y < 10$ and x = 3 (region A). On the other hand, instances with high value for all features except MET, will be mapped to centroids between $20 < y < 30$ and x = 23 (region B). Thus, from this example it shows that each feature has different significant in different region of the SOM map. Hence subspace clustering occurs in the SOM map.

In addition, the SOM algorithm can duplicate and at the same time simplify the abstract feature space geometry in the original dataset. This ability is labelled as 'topology-preserving' and has been extensively studied by Amerijckx et al., (1998), Ferreira et al., (2001), Vesanto and Alhoniemi (2000), and Villmann et al., (1997). In other words, the SOM could recreate and simplify the subspace(s) that exist in the training dataset.

A careful reader will realize that there is a contradiction in the above paragraph, when the SOM model is said to be Topology preserving, but also a subspace simplifier. In the author's opinion, the SOM model does not entirely preserve the original topology, but it simplifies the topology without losing the information that is stored in the original topology. This agree with Kohonen's description of SOM being a vector quantization algorithm.

5.5 SOM Model Optimization

Kohonen further described SOM as a projection mapping similar to vector quantization with the addition of being spatially globally ordered (Kohonen, 1982). Vector quantization is a data compression technique that reduces the number of bits required to represent information. However, vector quantization is a lossy data compression technique since a small portion of information can be lost in the process.

In the author's opinion, a SOM model is consider to be optimized when the loss of information is minimized. Loss of information can be reduced by configuring the SOM model with the correct hyperparameters for a given dataset. In this subchapter, two aspects of the SOM model had been measured to ensure that the model produced is the best model in representing the information stored in the original dataset:

1. The first aspect is to compare the general distribution of instances in a feature space volume created in a SOM model relative to the original dataset feature space volume. This can plausibly to be measured by using an equation that measures the determinant between two covariance matrices of two feature space volumes (Ho, & Bernadó-Mansilla, 2006). Such an equation is the modified Multivariate

Bhattacharyya Distance (MB-Distance), as given in equation (5.8) (Hong Y. et al., 2015);

2. The second aspect is to gauge the loss of information (entropy divergence) in the SOM model relative to the information available in the original dataset. This was measured by using the g (KL-Distance) (5.10, taken from Pardo L., (2005)), as this equation is a well-known method for measuring information loss in encoding.

$$D_{MB} = \left\| \frac{1}{8}(\boldsymbol{\mu_s} - \boldsymbol{\mu_c})^T \Sigma^{-1} (\boldsymbol{\mu_s} - \boldsymbol{\mu_c}) \right\| + \frac{1}{2} \ln \left(\frac{\det \Sigma}{\sqrt{\det \Sigma_s - \det \Sigma_c}} \right) \tag{5.8}$$

$$\Sigma = \frac{\Sigma_s + \Sigma_c}{2} \tag{5.9}$$

$$D_{KL} = \frac{1}{2} \left(tr(\Sigma_c^{-1} \Sigma_s) + (\boldsymbol{\mu_c} - \boldsymbol{\mu_s})^T \Sigma_c^{-1} (\boldsymbol{\mu_c} - \boldsymbol{\mu_s}) - k + \ln \left(\frac{\det \Sigma_c}{\det \Sigma_s} \right) \right) \tag{5.10}$$

Where $\boldsymbol{\mu}$, Σ **,** and *k* are the means, covariant matrix, and feature dimension respectively, while ***s*** and ***c*** denote the training dataset and the SOM map distribution respectively. For both these equations, the smaller the distance (D_{MB} ,D_{KL}) obtained, the more optimized the SOM model would become.

As stated before, this research employed three kinds of datasets, SUSY, Higgs, and Dimuon datasets. The SOM modes for the SUSY and the Higgs datasets have relatively fewer centroids (30 x 30 centroids) than the Dimuon dataset (100 x 100). Hence, both SUSY and Higgs SOM models had a relatively shorter training time than the Dimuon SOM model. The short training time allows the multiple SOM model to be created with different hyperparameter configurations in an acceptable length of time.

On the other hand, the Dimuon SOM model required a training time that was approximately twice as long as the Higgs and the SUSY SOM model training time. The

long training time inhibits the production of multiple Dimuon SOM models with different hyperparameters. Hence, the most optimized version of the Dimuon SOM model was not investigated.

5.5.1 Results for Feature Engineering

Figure 5.17 – **Figure 5.22** show the box and whisker plot of the recorded MB-Distance and KL-Distance for both the Higgs SOM and the SUSY SOM. The top and bottom of the box is the first (Q1) and third quartile (Q2) of the distribution while the red line inside the box represent the mean. The whisker at the top is equal to $Q3 + 1.5\ (Q3 - Q1)$ whereas the whisker at the bottom is equal to $Q1 - 1.5\ (Q3 - Q1)$. Cross above and below this whisker are outlier.

Higgs Dataset: In this subchapter, the effect of cherry-picking the feature for the SOM algorithm to module is presented. The MB-Distance and KL-Distance for training the SOM with different feature combinations are shown in **Figure 5.17**:

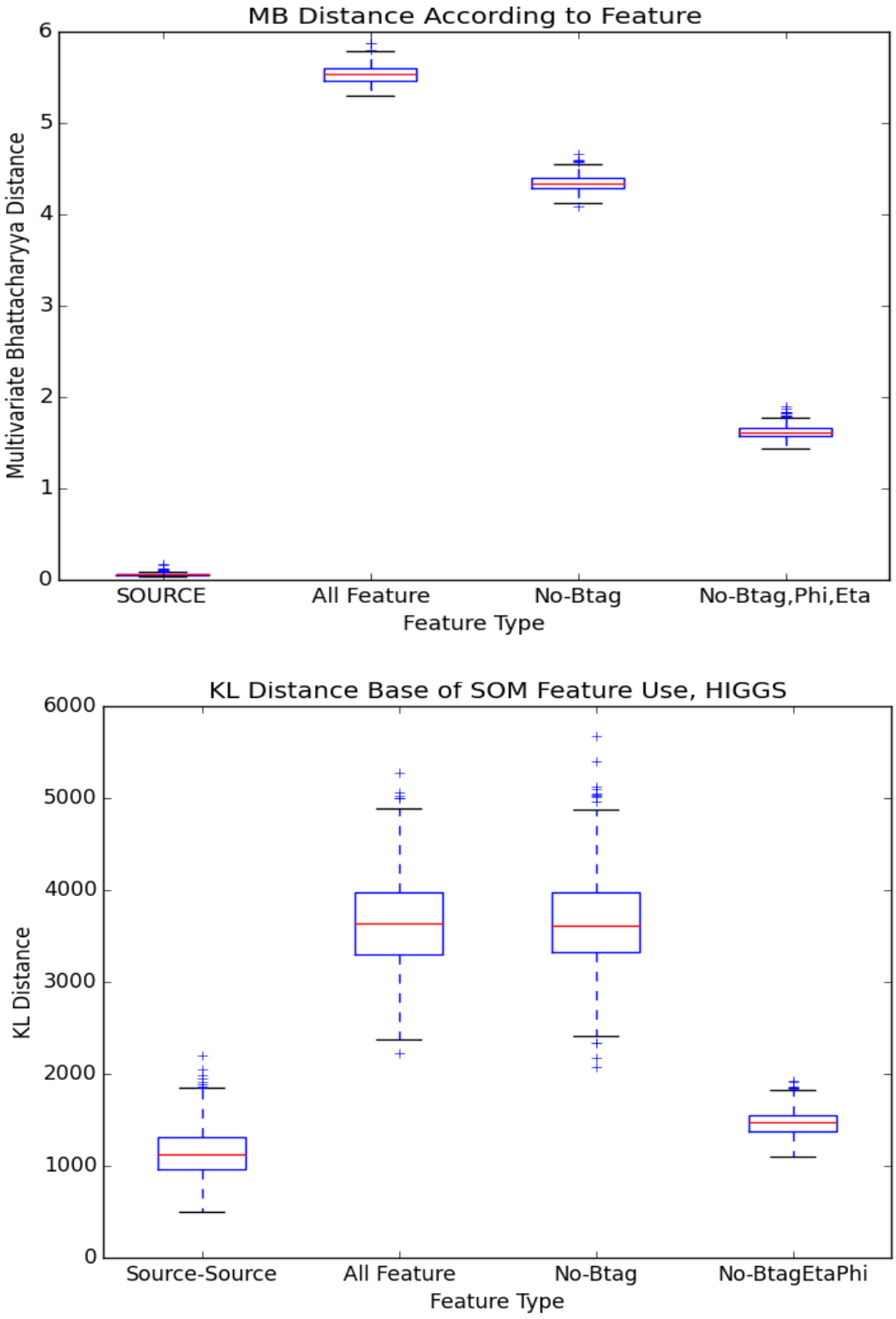


Figure 5.17: The MB-distance and the KL-Distance for the SOM model that had been trained by using various feature selections for the Higgs dataset.

Supersymmetry Dataset:

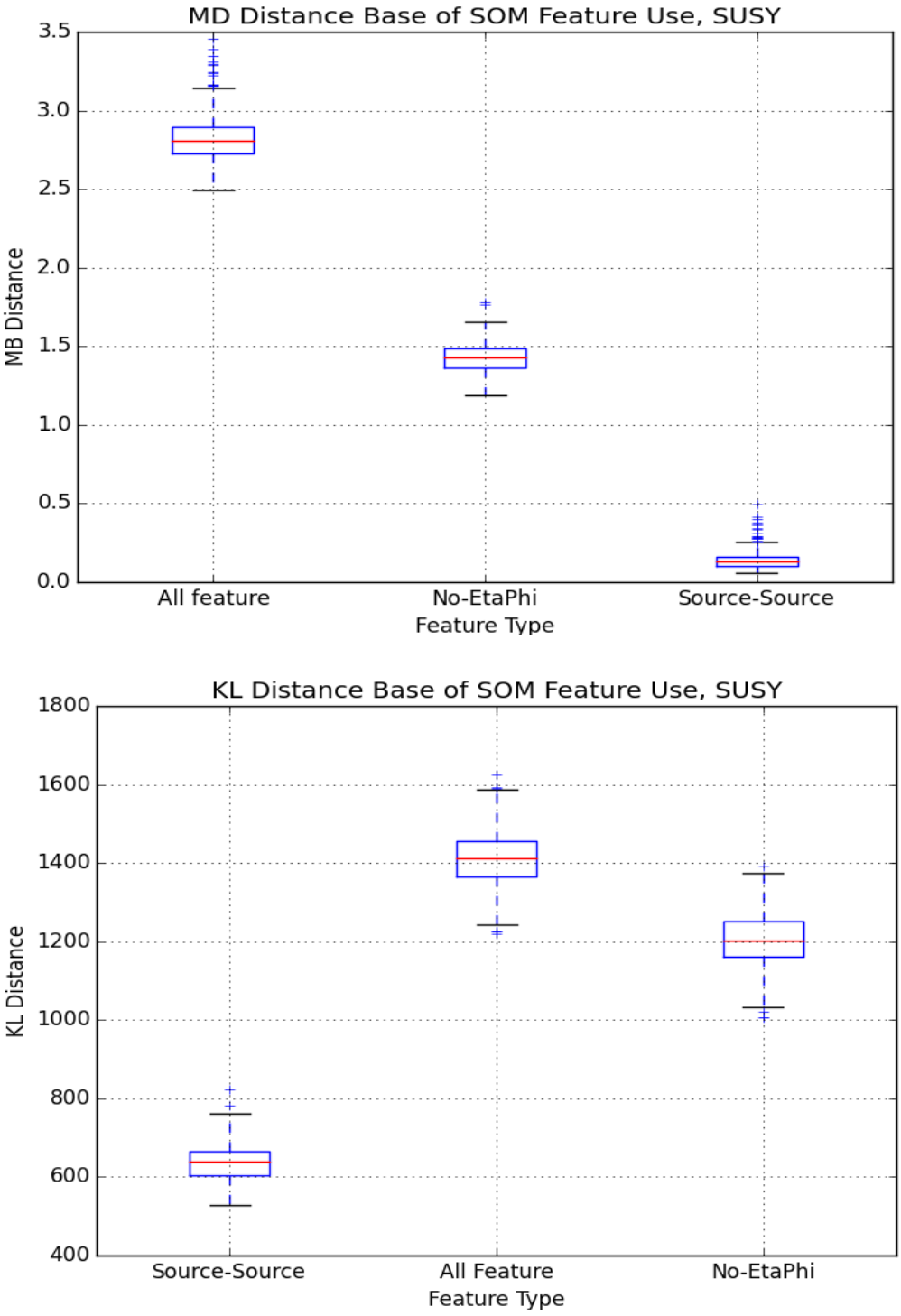


Figure 5.18: The MB-distance and the KL-Distance for the SOM model that had been trained by using various feature selections for the SUSY dataset.

5.5.2 Results for Different Training Iterations

Higgs Dataset: The results obtained concerning the MB and KL distances for training the SOM with different training iterations are given in **Figure 5.19** for the Higgs dataset:

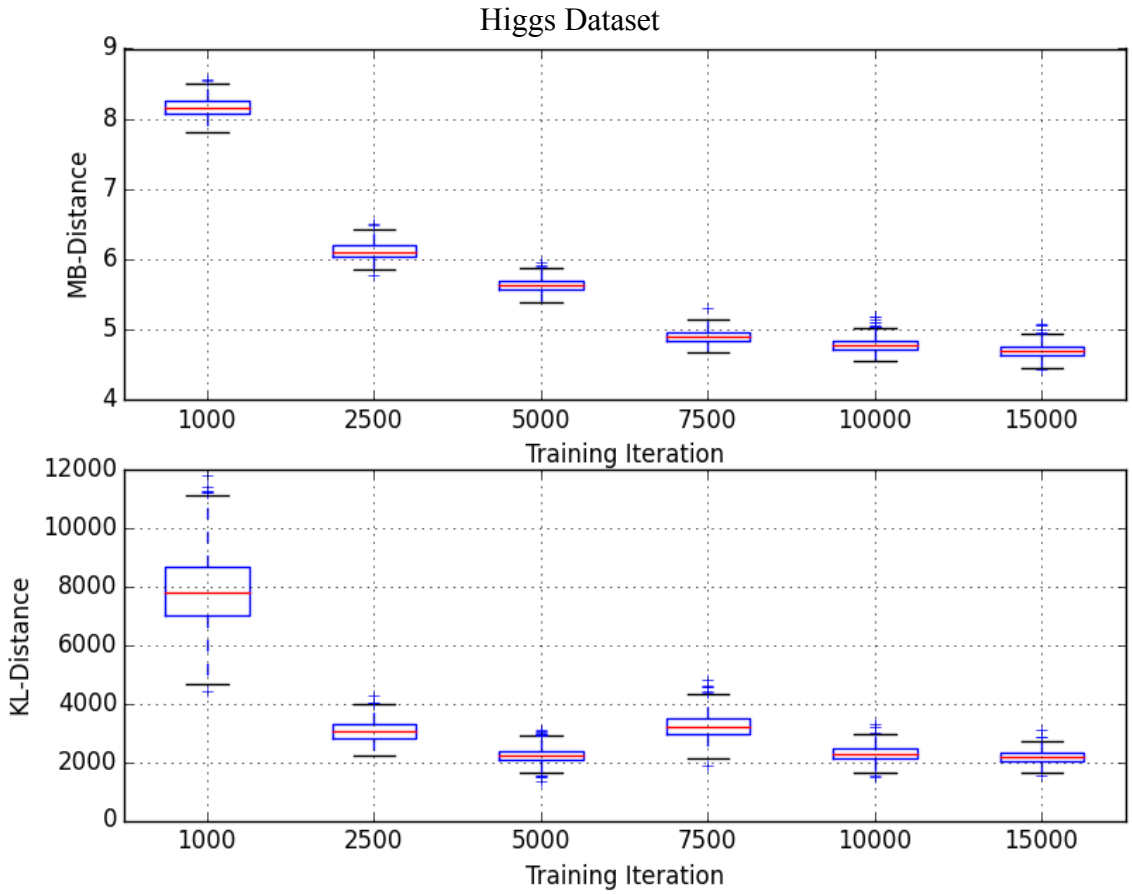


Figure 5.19: The MB-distance and the KL-Distance for SOM model that had been trained by using various training iterations for the Higgs dataset.

Supersymmetry Dataset: The results concerning MB and KL distances for training SOM with different training iterations are given in **Figure 5.20** for the Higgs dataset:

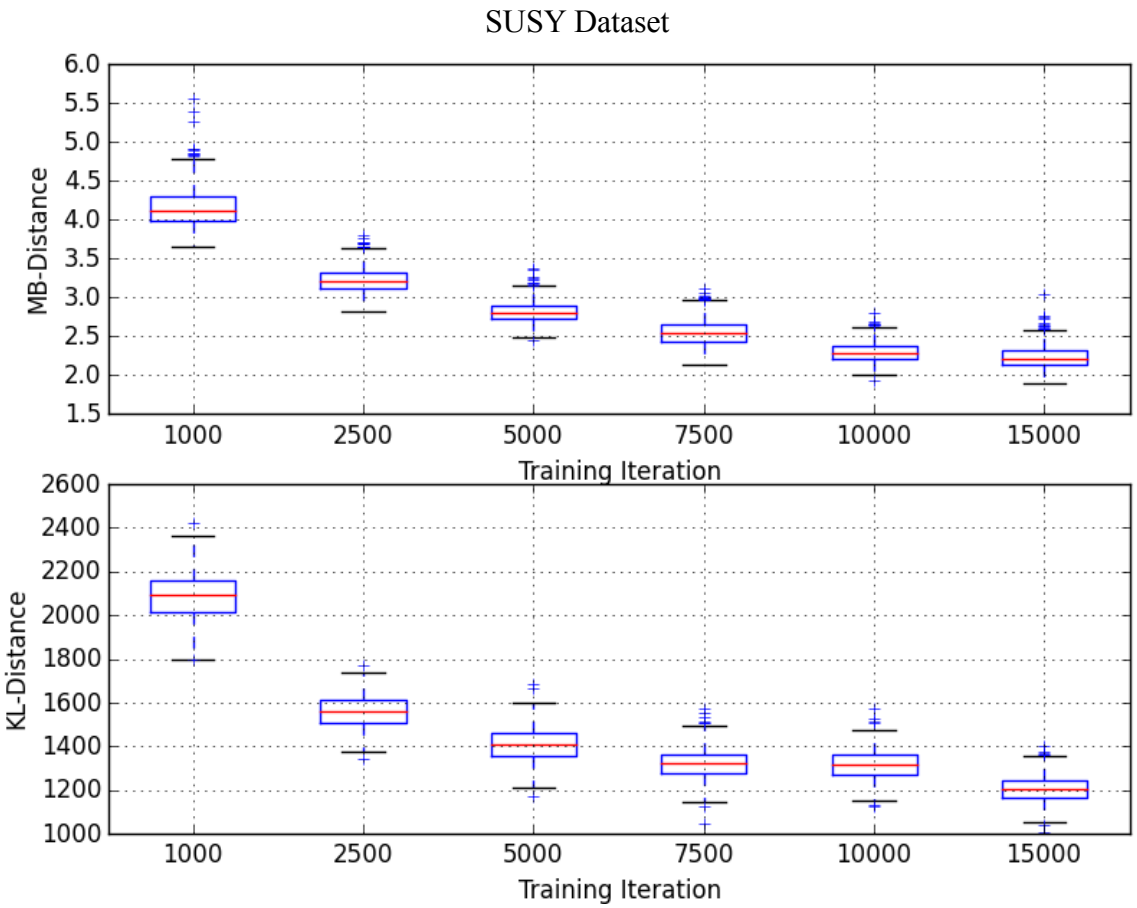


Figure 5.20: The MB-distance and the KL-Distance for SOM model that had been trained by using various training iterations for the SUSY dataset.

5.5.3 Results for Different Learning-Rate Function Iterations

Higgs Dataset: The results obtained for MB and KL distances in training SOM with different learning-rate functions are given in **Figure 5.21** for the Higgs dataset. The learning-rate functions that had been studied were derivative hyperbolic tan (Dtanh), reverse logistic regression (Revlogis), damped sinusoidal wave (Dsin), and logistic regression (Logis) on the Homogenous (glb) and the heterogeneous (lcl) modes, as described in subchapters 5.1.3 and 5.1.4:

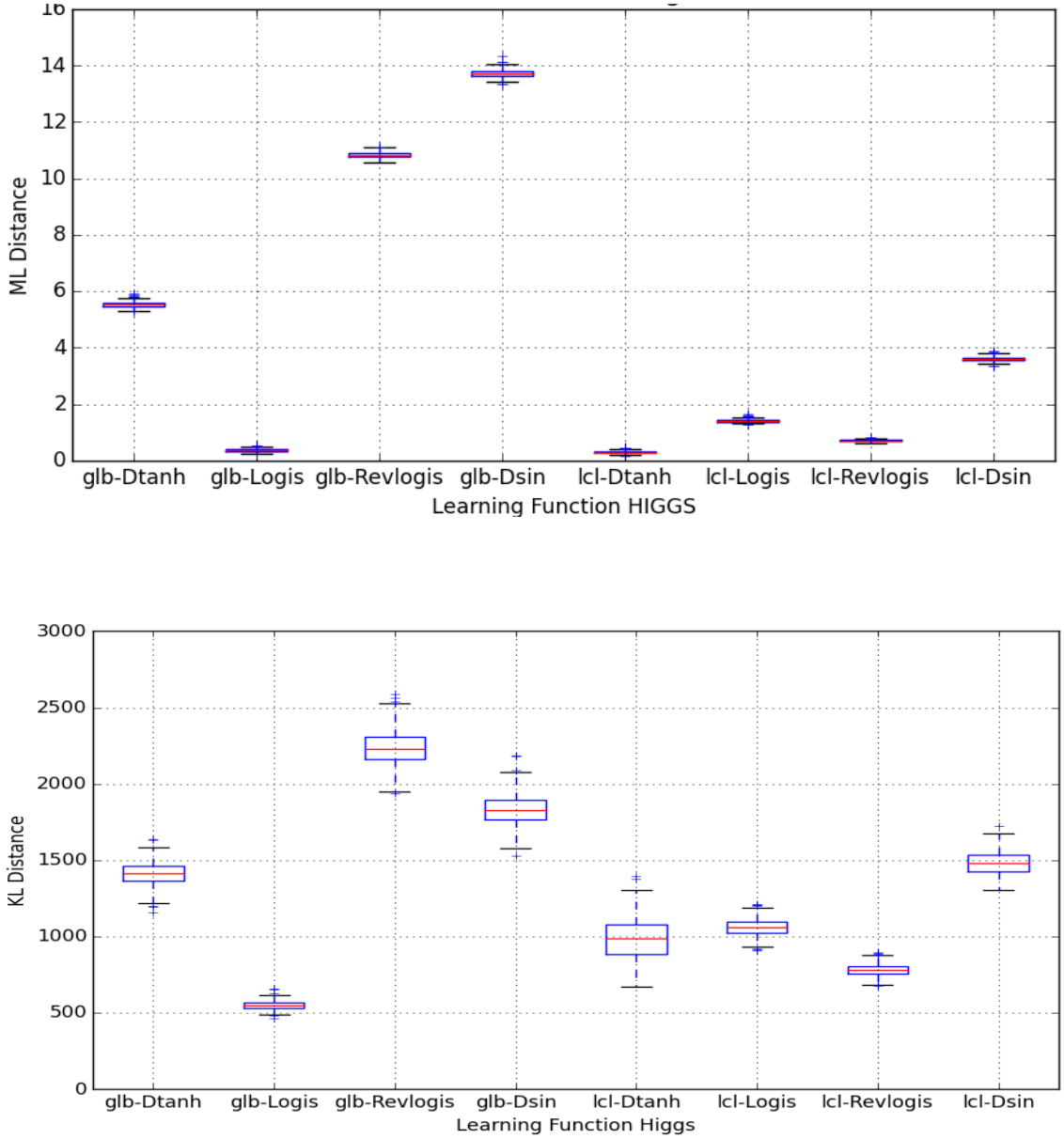


Figure 5.21: The MB-distance and the KL-Distance for SOM model that had been trained by using various learning-rate functions with the indicate modes for Higgs dataset.

Supersymmetry Dataset: The results obtained for MB and KL distances after training SOM with different learning-rate functions are given in **Figure 5.22** for the SUSY dataset:

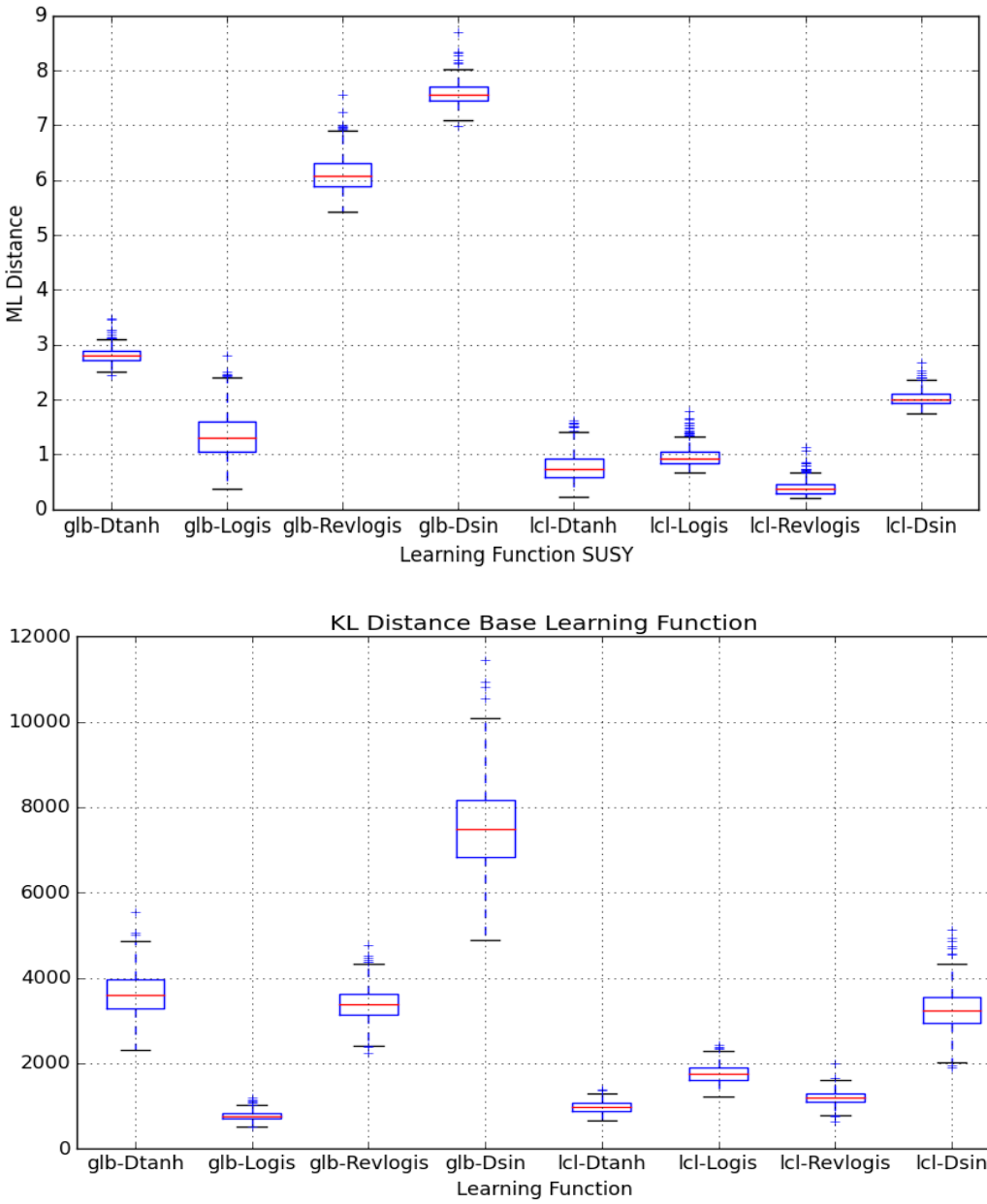


Figure 5.22: The MB-distance and the KL-Distance for SOM model that had been trained by using various learning-rate functions with the indicate modes for Higgs dataset.

5.5.4 Discussion on SOM Model Optimization based on MB and KL Distances

Higgs Dataset: The SOM model for the Higgs dataset had been used for classification purposes; this required the model to represent the original instance vector in the dataset with minimal divergence. Hence, the SOM model is optimized when the difference between the instance and the centroid weight-vector is minimum; therefore, the reduction in Kullback-Leibler divergence (distance) is greater importance than the MB distance.

In line with this, the SOM model was optimized better when it was developed without the Eta and the Phi feature types. The removal of these features from the model creation reduced the KL Distance by a factor of almost 2, from the original value of approximately 4.5k to approximately 1.5k, as shown in **Figure 5.17 (Right)**. On the other hand, inclusion or exclusion of B-tag types into the SOM modelling did not produce any noticeable changes in the KL-Distance. Thus, it should be kept to preserve information store in this feature.

Training iteration above 7,500 steps did not produce any significant reduction in the KL distance and the MB distance, as illustrated in **Figure 5.19**. The logistic regression learning-rate on Homogenous mode (glb-logis) generated the SOM with the lowest KL-Distance, as shown in **Figure 5.21.** Hence, the training iteration should be done at 7500 iterations with glb-logis as the learning-rate function.

Supersymmetry dataset: The SOM model that was developed by using the supersymmetry dataset will be used in the unsupervised clustering analysis. In this situation, the multivariate Bhattacharyya Distance is more crucial than the KL distance,

since the clustering algorithm analyses the location of instances in the SOM feature space to create clusters.

In this regard, the SOM model created without the Eta and the Phi features displayed a reduced MB distance of approximately 1.4, whereas the original SOM (trained with all feature) had an MB-distance of approximately 2.6, as shown in **Figure 5.18**. Thus this features is remove from the SOM model training.

In addition, **Figure 5.20** shows that both the MB distance and the KL distance continued to drop as the training iteration increased. Hence, the SOM model should be trained at the maximum training iteration of 15,000. As for the learning-rate function, **Figure 5.22** shows that the SOM model trained with Reverse logistic regression at heterogeneous mode (lcl-revlogis) produced the lowest MB distance, so this function should be used.

5.6 SOM Hyperparameter Conclusion

The objective of this chapter was to determine the hyperparameter configuration that produced the most optimized SOM model for the Higgs and the SUSY datasets. The first half of this chapter provides the introduction to various SOM hyperparameters and the mathematical equations that of them. One of the common practices in optimizing an ML model is to do future engineering. However, it was found that the Higgs dataset did not have one single future or future combination that offered the highest dissimilarity measurement between the signal and the noise instances. This means that the classification of the Higgs dataset required an ML that can do subspace clustering, such as the SOM model.

The optimization level of the SOM model was measured by using the modified Multivariate Bhattacharyya Distance equation and the modified Multivariate Kullback-Leibler distance. By using these two quantities, it was found that the Higgs SOM model was optimized when it was trained by using the logistic regression learning-rate on Homogenous mode at 7,500 training iterations. On the other hand, the SUSY SOM model was optimized when it was trained by using the Reverse logistic regression at the heterogeneous mode at 15,000 training iterations. The optimized hyperparameters configuration determined in this chapter is further used in chapters 6 and 7.

CHAPTER 6
SOM FOR CLASSIFICATION

The ultimate test of any classification model is its performance. If discriminant analysis gives a satisfactory predictive power for nonnormal samples, don't let the rigor of theory stand in your way

(Narsky & Porter, 2013, p.226)

6.1 Chapter Introduction

The SOM algorithm was not designed to create a classification model, but instead a clustering model. However, due to the fact that SOM that can create subspace clustering, pairing an SOM model with another classification algorithm may enhance the classifier accuracy. This chapter therefore focus on developing two classification models by pairing Linear Discrimination Analysis (LDA) with SOM to create the SOM+LDA model, and pairing Quadratic Discrimination Analysis (QDA) together with SOM to create the SOM+QDA model. The classification results obtained from these two methods were compared to those of QDA only, LDA only, SVM and RF algorithms.

The details of the application used to create the SOM model are given in Chapter 4, while optimization of the SOM model is discussed in Chapter 5. The main hardware that was used to generate all the ML models in this chapter was the UM sifir cluster, previously discussed in Chapter 3. Details on QDA, LDA, SVM and RF algorithm has been provided in Chapter 4.

6.2 SOM-Q/LDA Higgs Dataset Classification Method

In this research, a 'stacking-model' model is created in which the Quadratic / Linear discrimination analysis, is employed together with the SOM to construct a classification model shown in **Figure 6.1**. In this method, the first step was to create and to train an SOM model based on the training dataset. Once the training was completed, instances from the training dataset were mapped back to the SOM model. The mapping process resulted in the creation of LIC on most of the centroids on the SOM map. Discussion regarding the SOM mapping method and the LIC are given in subchapter 4.5.3.

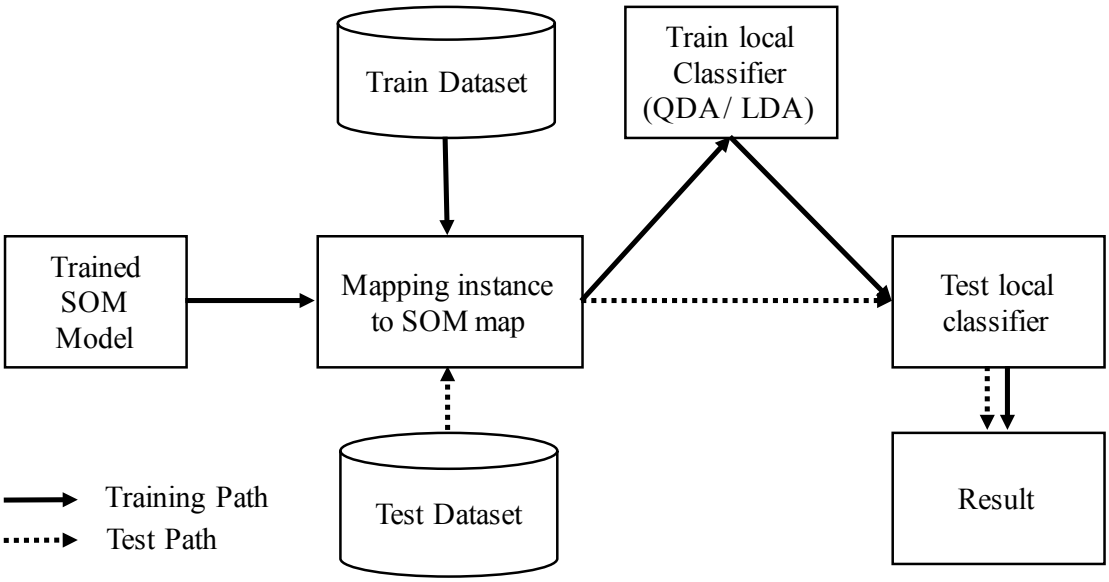


Figure 6.1: The adopted stacking model for classifying the Higgs dataset instances**.**

For each centroid with an LIC count of more than 50 instances, a local LDA/QDA model was created solely based on the LIC, thus, an LDA/QDA model that was unique

to each centroid on the SOM map was generated. To test the developed model, the instances from the test dataset were then mapped on the SOM model, and classified by the local LDA/QDA model. The classification results were compared with other classification algorithms.

6.2.1 Mapping Results

Subchapter 5.2.3 showed that the similarity functions that should be used for the Higgs dataset were Euclidean Distance, City-block, and Cosine function, due to that fact that these functions were better suited for a high-dimensional dataset. Additionally, it was also established in subchapter 5.3.5 that an optimized SOM model for the Higgs dataset could be created by using logistic regression learning-rate in a Homogenous mode with 7,500 training iterations.

Once the SOM model had been trained with this hyperparameter configuration, each instance from the dataset was mapped to a centroid with the most similar weight-vector, explained previously in subchapter 4.3.3. The similarity between instance and centroid was measured by the same similarity function that was used in the SOM model training, as discussed in subchapter 5.1.5. The distribution of instance count in each centroid LIC on the SOM map for all similarity functions is shown in **Figure 6.2.**

Figure 6.2 shows that instance number is varied between each centroid's LIC. These numbers were almost independent of which dataset was being mapped to the SOM map, since the dataset was randomised. Thus, the distribution of LIC count across the SOM map is reproducible for any given dataset.

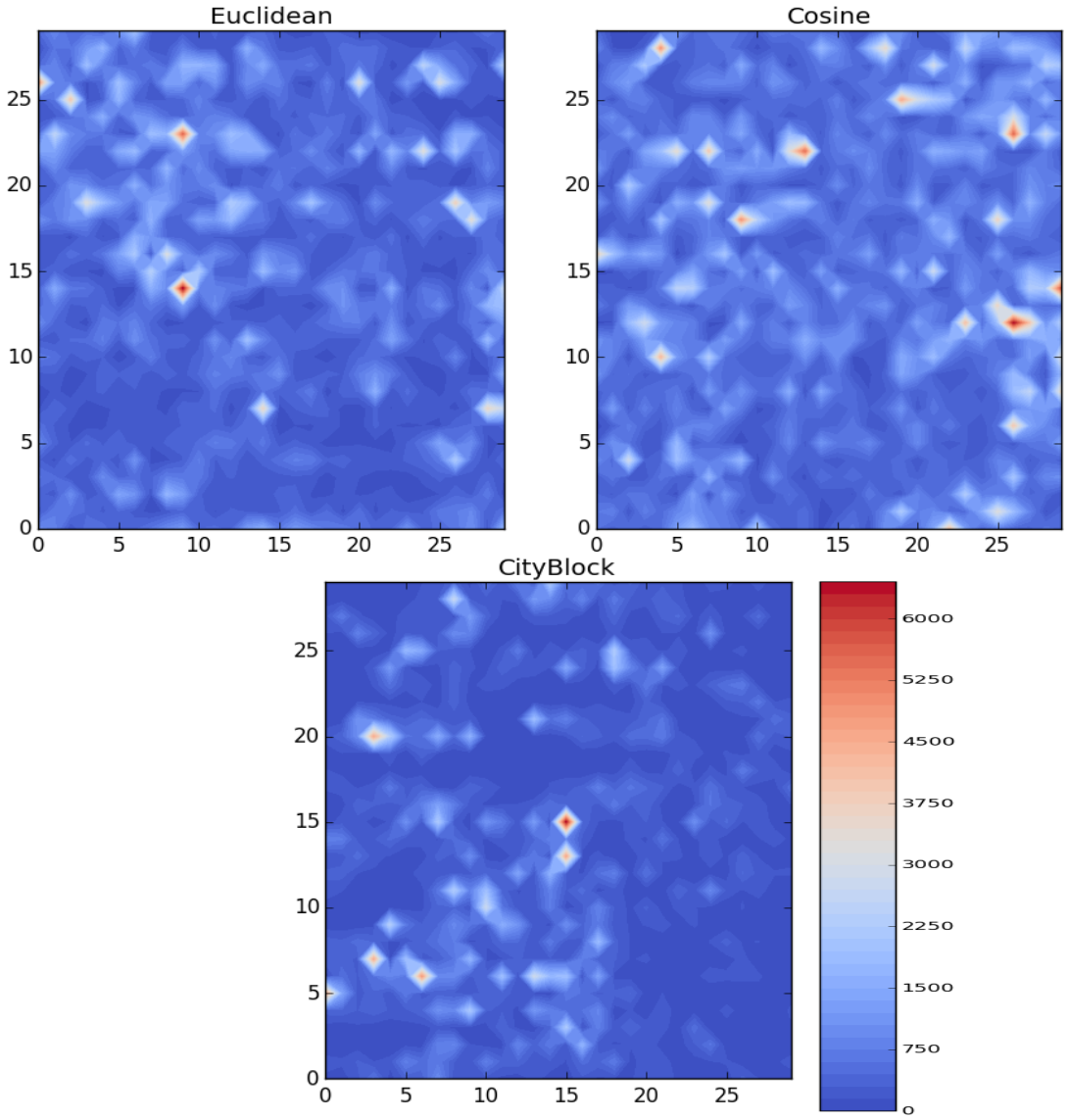


Figure 6.2: The distribution of LIC count across the SOM map that was created by different similarity functions. The colour-bar denotes the number of instances at each centroid.

To prove that the LIC count distribution is same for any given randomised Higgs-dataset, another Higgs-dataset (labelled as dummy-dataset) was mapped on to the previous SOM map. The difference in the LIC count between the training and the dummy dataset is shown in **Figure 6.3** and **Table 6.1;**

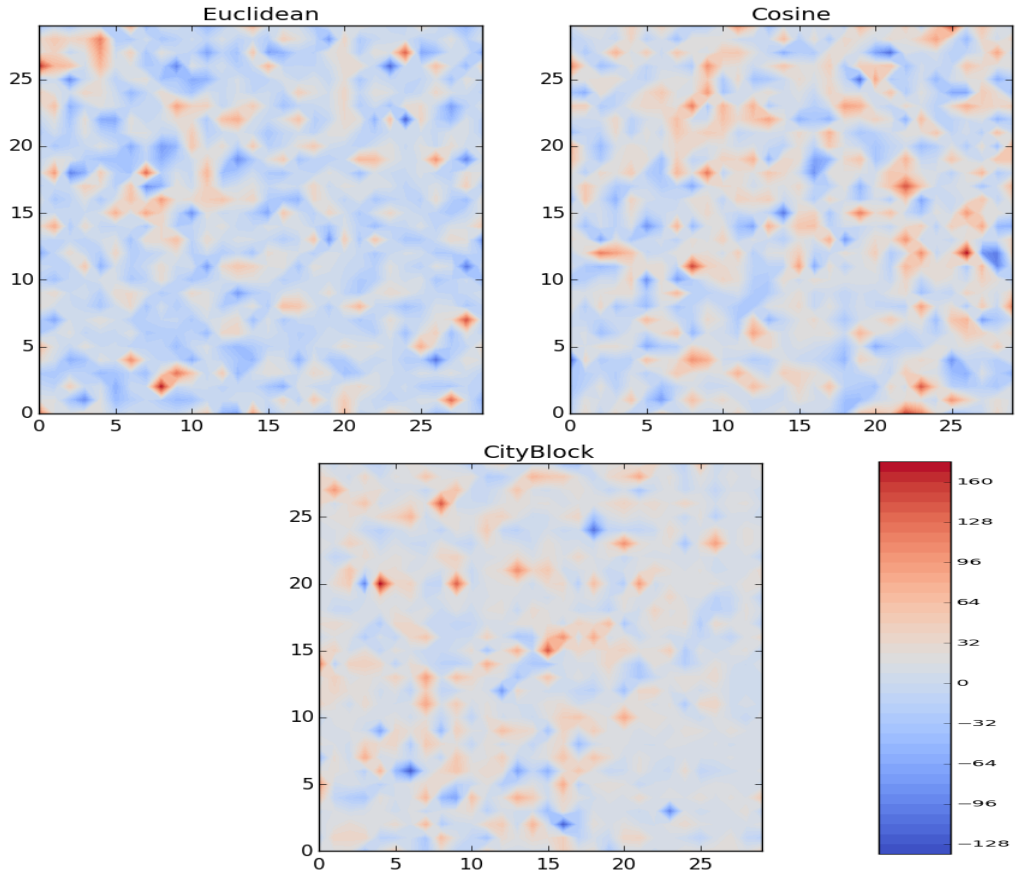


Figure 6.3: The difference in number of instances for each centroid LIC across the SOM map between the training dataset and the dummy dataset for each similarity function.

Table 6.1: Comparison of mean and standard deviation (STD) for instance number per centroid LIC between training and dummy datasets for each similarity function

Similarity Function	Training Dataset		Dummy Dataset		Difference	
	Mean	STD	Mean	STD	Mean	STD
Euclidean	555.556	618.018	555.556	621.086	0.000	33.857
City-block	555.556	964.332	555.556	964.929	0.000	32.831
Cosine	555.556	564.929	555.556	569.603	0.000	32.115

Figure 6.3 and **Table 6.1** show that there is no major difference in the LIC number for each centroid between the training and the dummy datasets. **Table 6.1** shows there is no difference in the average (mean) LIC count between the two SOM map and the different in the standard deviation is only around 0.5% which is negligible. Thus, it can be concluded that the LIC count across the SOM map (population density) was independent of the dataset being used, provided that it is a randomised dataset.

6.2.2 Local QDA and LDA Classification Results

One all the instances from the dataset has been mapped to the SOM map, any centroid with LIC count more than 49 will have an its own LDA/QDA. Each LDA/QDA classifier is trained using their own LIC.

For centroids with LIC count less than 50 does not have its own LDA/QDA, and instances that were mapped to these centroids were re-mapped to the next most similar centroid. Commonly, the number of rejected centroids would be less than 5% of the total centroids in the SOM model.

After each LDA/QDA classifier had been developed, all instance from the training and test dataset were classified by their respective centroid's LDA / QDA. The results from this local classification are shown in **Table 6.2** and **Table 6.3**; the results obtained here are used later to study the model variance and bias properties. With regard to **Tables 6.2** and **6.3**, the formula for accuracy and sensitivity are given by equations (6.2) and (6.3) respectively;

$$\text{Accuracy} = \frac{Tp + Tn}{Tp + Tn + Fn + Fp} \tag{6.2}$$

$$\text{Sensitivity} = \frac{Tp}{Tp + Fn} \tag{6.3}$$

where *Tp, Tn, Fn*, and *Fp* stand for true positive count, true negative count, false positive count, and false negative count, respectively.

Table 6.2: The results of SOM-LDA Classification for the Higgs training and test datasets.

SOM-Local-LDA Training Dataset					
Similarity Function	No. Local Classifier	Mean Accuracy	Mean STD	Mean Sensitivity	Sensitivity STD
Euclidean	896	0.730	0.066	0.693	0.080
City-block	854	0.738	0.075	0.700	0.085
Cosine	888	0.729	0.065	0.697	0.065
SOM-Local-LDA Test Dataset					
Euclidean	896	0.691	0.073	0.665	0.106
City-block	854	0.691	0.077	0.653	0.077
Cosine	888	0.688	0.072	0.667	0.108

Table 6.3: The results of SOM-QDA Classification for the Higgs training and test datasets.

SOM-Local-QDA Training Dataset					
Similarity Function	No. Local Classifier	Mean Accuracy	Mean STD	Mean Sensitivity	Sensitivity STD
Euclidean	896	0.653	0.158	0.651	0.294
City-block	854	0.686	0.166	0.689	0.280
Cosine	888	0.634	0.156	0.649	0.292
SOM-Local-QDA Test Dataset					
Euclidean	896	0.586	0.119	0.560	0.286
City-block	854	0.601	0.122	0.551	0.288
Cosine	888	0.578	0.118	0.567	0.289

Table 6.2 shows that the SOM+LDA model has a lower classification accuracy for the test dataset then the training dataset, with the difference in the score between 0.039-0.047. The sensitivity also decreases by 0.028-0.047. The reduction in score

suggest that the model has a slightly higher degree of variance over bias; however, the difference is so small that it can be neglected.

Table 6.3 show that the SOM+QDA accuracy and sensitivity score decreased when it is classifying the test dataset compare to the training dataset. The score for the test dataset is only just slightly better than a random classifier (score = 0.5). This result shows the SOM+QDA has a much higher variance then the SOM+LDA since its score drop when classifying test dataset.

6.2.3 Comparison with Other Classifiers

The Higgs test dataset contained 1,500,000 instances with signal to noise ratio of 1.1244. The classification results for support vector machine (SVM), random forest (RF), LDA, and QDA are given in **Table 6.4.** In the table, S.No., N.No., Tp, Tn, Fp, and Fn denote the signal instances number, noise instances number, true positive count, true negative count, false positive count, and accuracy. Furthermore, **Table 6.4** shows the higher metric evaluation on the same algorithm and datasets, with precession calculated by using the formula given in (6.4), while **Figure 6.5** shows the Receiver Operating Characteristic (ROC) plot.

$$\text{Precession} = \frac{Tp}{Fp + Tp} \tag{6.4}$$

Table 6.4: Comparison of results between the different types of classification algorithms for the Higgs dataset

Algorithm	Tp / S.No.	Tn / N. No.	Fp / N.No.	Fn / S.No.
SVM	0.795	0.608	0.392	0.205
RF	0.719	0.747	0.253	0.281
LDA	0.664	0.606	0.394	0.336
QDA	0.802	0.465	0.535	0.198
SOM+LDA				
Euclidean	0.708	0.679	0.321	0.292
City-block	0.707	0.679	0.321	0.293
Cosine	0.706	0.682	0.318	0.294
SOM+QDA				
Euclidean	0.551	0.496	0.416	0.370
City-block	0.522	0.532	0.369	0.398
Cosine	0.532	0.516	0.398	0.384

Table 6.5: Comparison of results between different typed of classification algorithms for the Higgs dataset using higher evaluation metric.

Algorithm	Accuracy	Precession	Sensitivity	ROC -AUC
SVM	0.707	0.695	0.795	0.777
RF	0.732	0.761	0.719	0.814
LDA	0.637	0.655	0.664	0.684
QDA	0.644	0.628	0.802	0.712
SOM+LDA				
Euclidean	0.694	0.713	0.708	0.757
City-block	0.694	0.712	0.707	0.756
Cosine	0.695	0.714	0.706	0.758
SOM+QDA				
Euclidean	0.573	0.598	0.598	0.554
City-block	0.578	0.614	0.568	0.564
Cosine	0.573	0.600	0.581	0.553

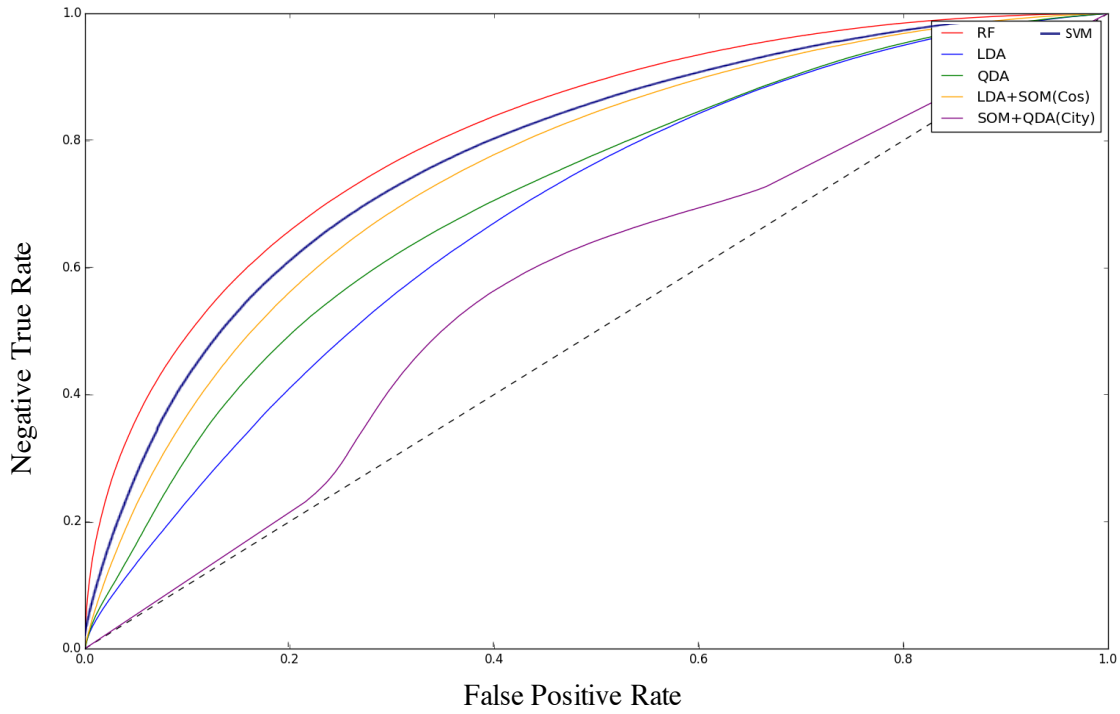


Figure 6.4: The ROC curve of various classification algorithms for the Higgs dataset.

6.2.4 Classification Results Discussion

Table 6.5 and **Figure 6.4** show that the random forest was the best classifier for the Higgs dataset, as the algorithm recorded the highest score in accuracy, precession, and ROC-Area Under the Curve (ROC-AUC). Meanwhile, SVM displayed the highest sensitivity, followed by RF, and SOM+LDA. The SOM+LDA only had a higher precision score compared to SVM, thought it was the third highest in all other categories. The QDA result was ignored since the results depicted in **Table 6.5** show that the algorithm label almost all instance as a signal.

What is more important to be highlighted here is that the LDA performance had improved when coupled with SOM. For example, the cosine SOM+LDA model improved its performance when compare to the LDA model without SOM by +9.09%, +9.03%,

+6.36%, and +10.83% for accuracy, precision, sensitivity, and ROC-AUC values respectively. On the other hand, the SOM did not improve the result for the QDA model, as shown in **Tables 6.4** and **Table 6.5.**

Furthermore, **Tables 6.4** and **Table 6.5** also demonstrate that changing the similarity function that was used to create the SOM map did not have any significant impact on the classification score of the model. This fact is true for both the SOM+LDA and the SOM+QDA models.

6.2.5 Role of SOM in LDA Classification

The ROC plot in **Figure 6.4** and the ROC-AUC value that is tabulated in **Table 6.5** show that the SOM model improved the classification capability of LDA; from a weaker classifier to a stronger classifier, with classification score almost equal to the SVM. Since the SOM+LDA with cosine similarity function gave the highest AUC score in **Table 6.5**, the following discussion is based solely on the result of the cosine SOM model.

The enhancement provided by the SOM in the SOM+LDA model is derived from its capability to recluster a large dataset into multiple LIC, which was more easy for the LDA to be classify.

As stated in **Table 6.3**, the Cosine SOM+LDA model had a total of 888 local LDA classifiers for each selected centroid on the SOM map. **Figure 6.5** shows each of these 888 local LDA classifiers plotted in a 3-Dimensional scatter plot. Two of the plot axes indicate the training dataset LIC count and its purity for a given LDA classifier. The third

axis indicates the accuracy in classifying LIC from the test dataset. **Figure 6.6** represents the same information in a 2-Dimensional representation for better visualization. The local instance purity for each centroid cluster is given by equation (6.5).

$$\text{Purity} = \frac{Local\ Signal\ Count - Local\ Noise\ Count}{Local\ Instance\ Count} \tag{6.5}$$

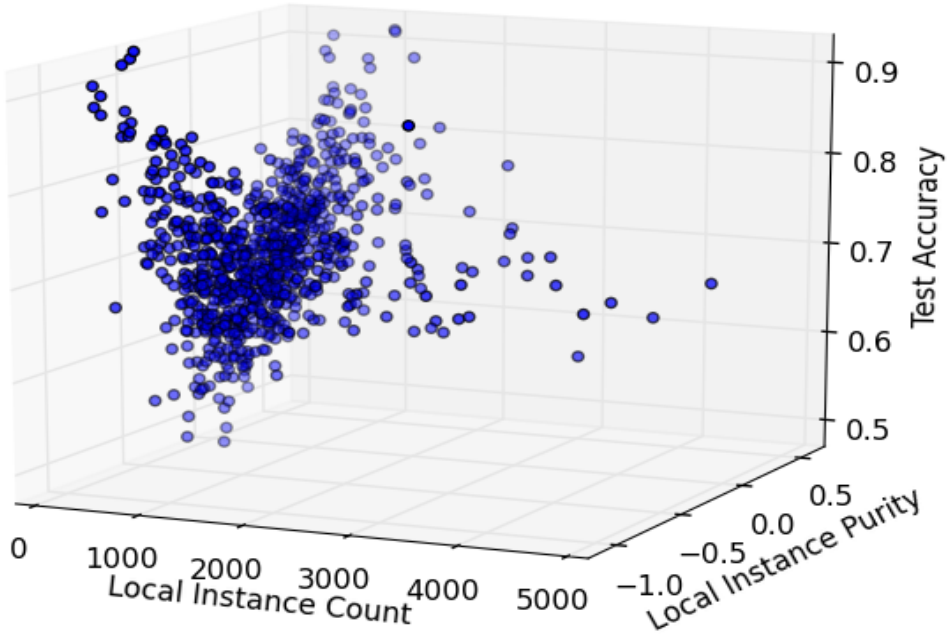


Figure 6.5: The scatter of each local SOM+LDA classifier based on the model local training instance count, local instance purity, and consequence test accuracy.

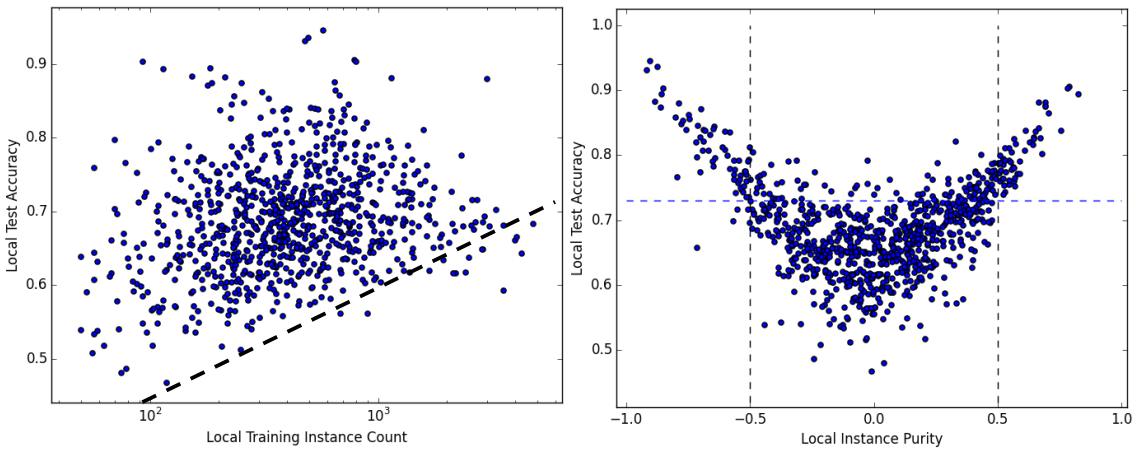


Figure 6.6: (Left) Scattering of local classifier based on their local training instance count and subsequent test accuracy, while (Right) is the scattering of local classifier based on the local instances purity and subsequent test accuracy.

The dashed line in **Figure 6.6** (Left) shows that when the number of local training instances increased, the minimal test accuracy also increased. Thus, the accuracy of classifying the Higgs instances (test dataset) has some correlation with the number of training instances that were used to train the local LDA model.

On the other hand, **Figure 6.6** (Right) shows that the test accuracy also had a strong correlation with the purity of the LIC. The majority of the local classifiers had a test accuracy of more than 0.73 (above the blue dashed line) when the training LIC purity was greater than 0.5 (more signal) or lower than -0.5 (more noise). Other than that, the local classifier with instance purity between -0.5 and 0.5 had a test score mostly below 0.73 (below the blue dashed line), similar to the accuracy obtained for normal LDA without SOM.

From the above, it can be concluded that the test accuracy score increased as the number of local training instances increased and/or the local training instances had high absolute purity.

For any given centroid, the purity of a local instances cluster had no detectable correlation with the local instances count, although a majority of the LIC with high purity had low LIC number (less than 100). This condition is shown in **Figure 6.7** where the 2-

dimnesional histogram shows the number of centroids with a given LIC purity and LIC number (local training instance count) for different similarity functions.

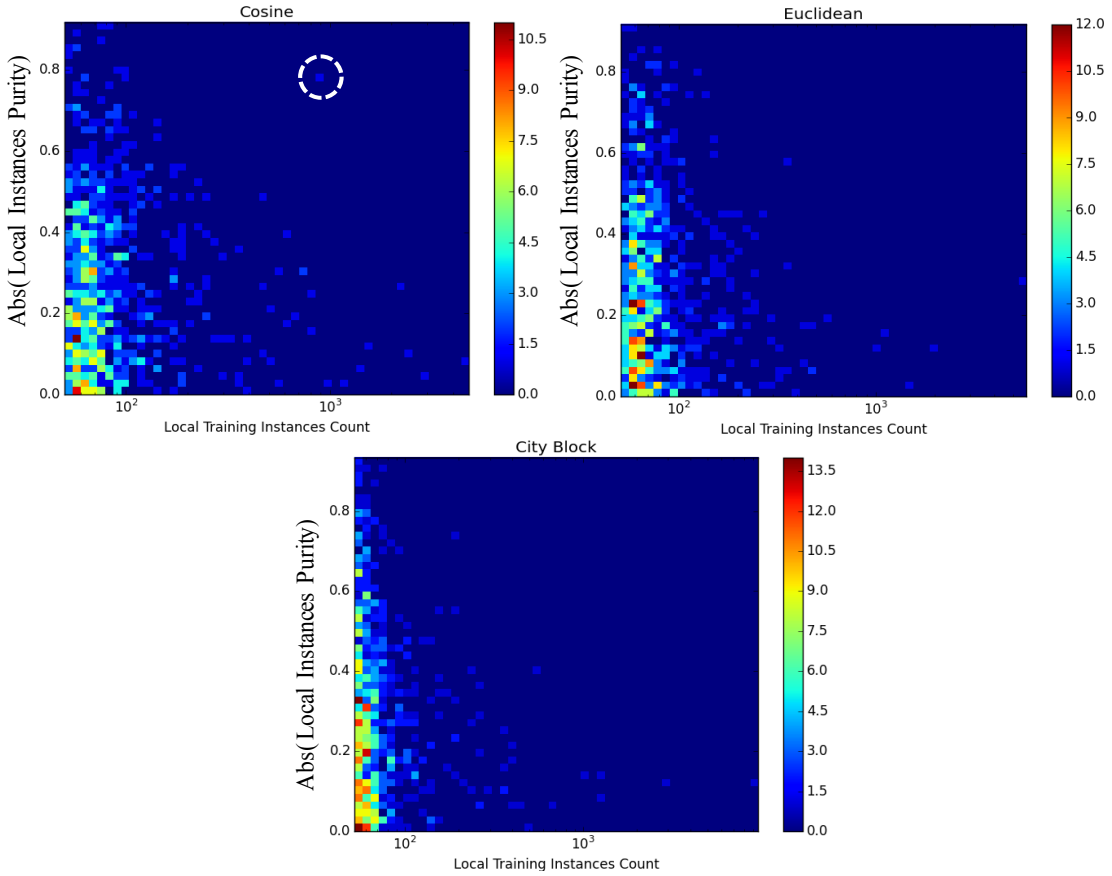


Figure 6.7: The 2-Dimensional Histogram of Absolute Instance purity versus Training Instance Count for different similarity functions. The circled region portrays high-level of purity and instance count.

In this research, it was found that the similarity function used to train and map the SOM model affected the number of centroids with high local instance purity (||Purity|| ≥ 0.8). This is shown by a cumulative histogram of the LIC purity for each similarity function SOM model as shown in **Figure 6.8**. The figure indicates that the Euclidean SOM model exhibited higher cumulative purity distribution over other similarity functions; however, it still had a lower AUC score (**Table 6.5**) than the SOM model with

Cosine similarity. This is due to the fact the SMO-Cosine model had centroid with a high number of instances and purity, as encircled in **Figure 6.7**.

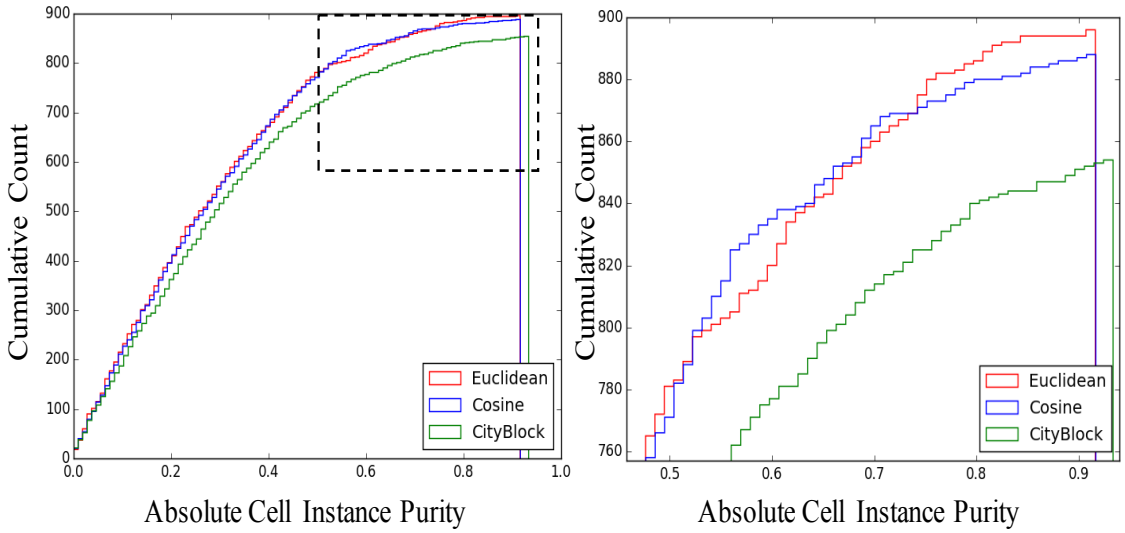


Figure 6.8: (Left) The cumulative count for each local instance purity for different SOM similarity functions, (Right) the zoomed plot for region in the dashed line box.

6.2.6 SOM+LDA Classification Conclusion

The following conclusion can be made regarding the application of SOM+LDA for Higgs instance classification;

1. SOM improved the LDA classification results by creating small numbers of local clusters with high instance purity, in which the LDA essentially did not play any classification role since a majority (> 80%) of the instances already

belonged to a certain class (noise or signal class). The clusters with ||Purity|| ≥ 0.50 had 0.809 accuracy, 17.2% higher than LDA alone.

2. In this research, it was found that the number of LIC with a high-level of purity depends on which similarity function was used to train and map the SOM model.

6.3 Cluster Purity Analysis

It had been proven that one of the factors that gave the SOM+LDA model a higher classification accuracy than the LDA model is due to the SOM ability to produce LIC with high purity for noise or signal instances. The LIC purity, in turn, is determined by the centroid weight-vectors that were used in the mapping process. This situation is shown in **Figure 6.9**, which includes the distribution of several features (a, b, c) as well as the resulting LIC purity (d) across the same Euclidean SOM map.

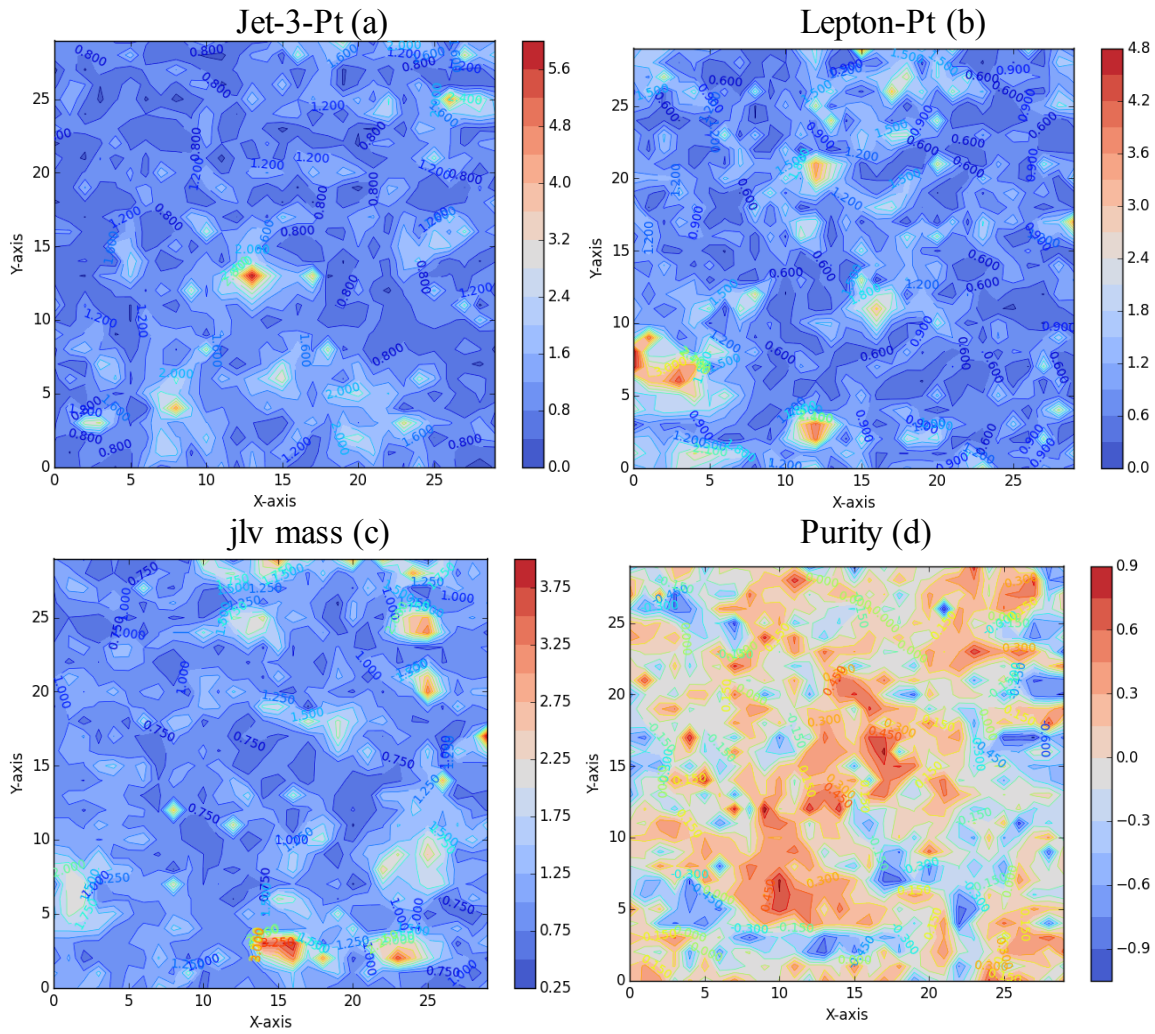


Figure 6.9: The distribution of Jet-3-Pt (a), Lepton-Pt (b), and jlv mass (c) across the Euclidean SOM map with the purity shown (d).

6.3.1 Correlation Between Weight-vector and Purity

A simple study was done to examine if a linear correlation exists between the weight-vector of an LIC and the purity it has. Table 6.7 show the correlation between the weight-vectors and the purity for a given feature for two SOM models. As shown in **Table 6.7**, no feature had a positive correlation with the LIC purity, on the other hand, the Lepton pt, m_{wbb}, MET, m_{wwbb}, and m_{bb} feature had a considerably more negative correlation with the LIC purity.

Table 6.7: The correlation between the weight-vector value for a particular feature and the LIC purity for Euclidean and Cosine SOM.

Feature	Euclidean	Cosine
jet4 pt	0.102	0.053
jet1 pt	0.097	0.149
jet2 pt	0.072	0.057
jet3 pt	0.053	0.031
m_{jj}	0.053	0.009
m_{jjj}	0.048	0.023
m_{lv}	0.009	0.011
jet4b-tag	-0.012	0.011
jet3b-tag	-0.038	-0.061
jet1b-tag	-0.070	-0.041
jet2b-tag	-0.092	-0.068
m_{jlv}	-0.141	-0.075
Lepton pt	-0.213	-0.224
m w_{bb}	-0.233	-0.280
MET	-0.270	-0.218
m_{wwbb}	-0.331	-0.367
m_{bb}	-0.416	-0.396

Figure 6.10 and **6.11** show a 2-dimensional histogram of the LIC purity versus the value of a particular feature for Euclidean and Cosine SOM models, respectively. Only the three features with the lowest correlation is shown in **Figure 6.10** and **Figure 6.11** as comparison to other randomly pick feature that has more positive correlations.

Figure 6.10 shows that the LIC purity declined as the magnitude of m_{bb}, m_{wwbb}, and MET increased. Whereas the jet 4 pt, m_{jj}, and m_{lv} did not have any correlation with the LIC purity. The figure also shows that LIC with high number of purity had a tendency to have the magnitude (of m_{bb} mass, m_{wwbb}, and MET) in a specific interval; around ~1 for m_{bb} and m_{wwbb}, between 0.5 -1.5 for MET. The m_{lv} mass plot is narrow because majority of the instance in the training dataset has value of ~1.

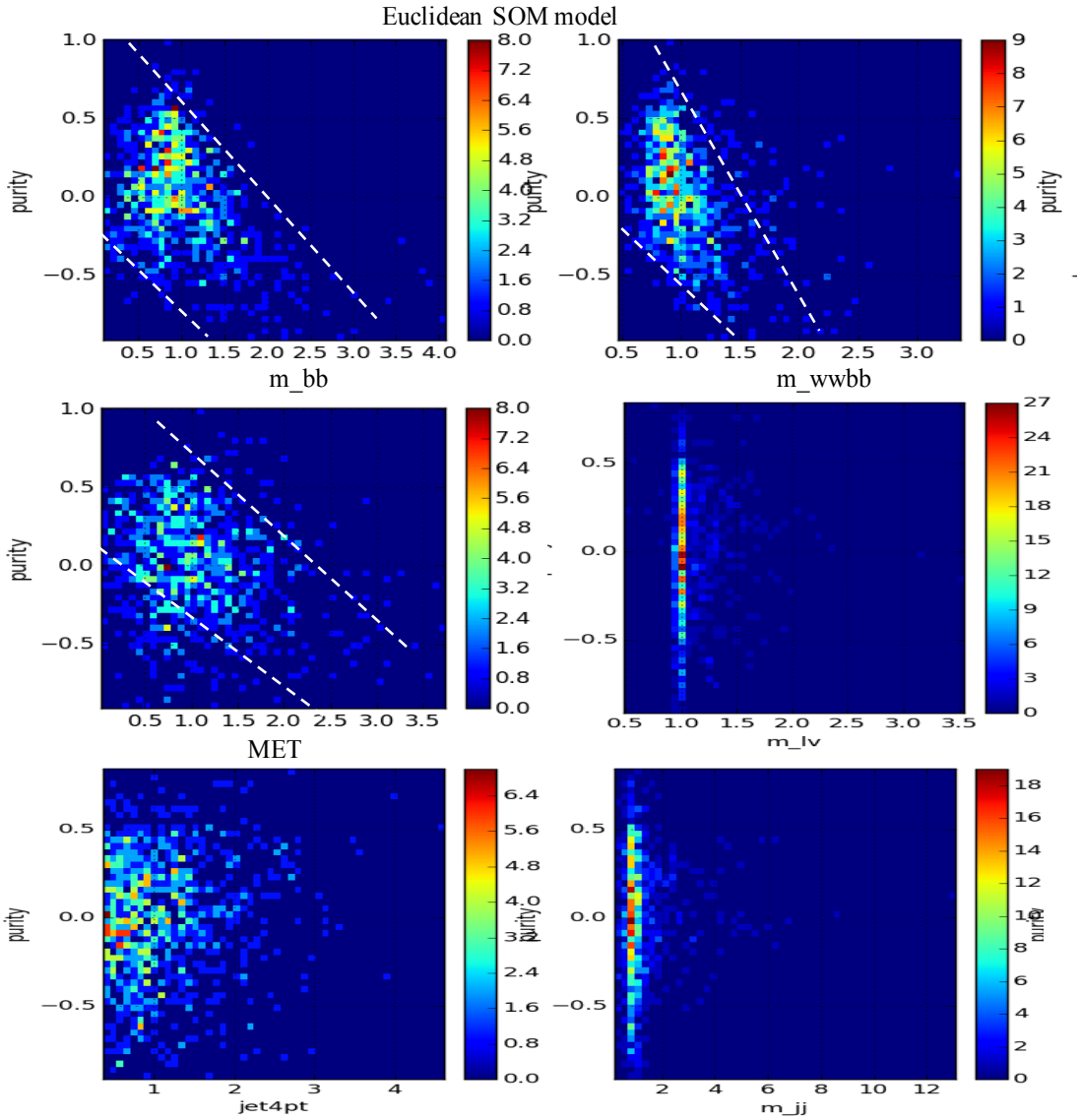


Figure 6.10: The 2-dimensional histogram of LIC purity versus with indicated feature for the Euclidean SOM model. Where shown, the dotted lines indicate the correlation between purity and the value of the variable.

The same trend is visible in the Cosine SOM model, as shown in **Figure 6.11**, with the addition that high purity LICs tended to have m_{wbb} also at around 1.

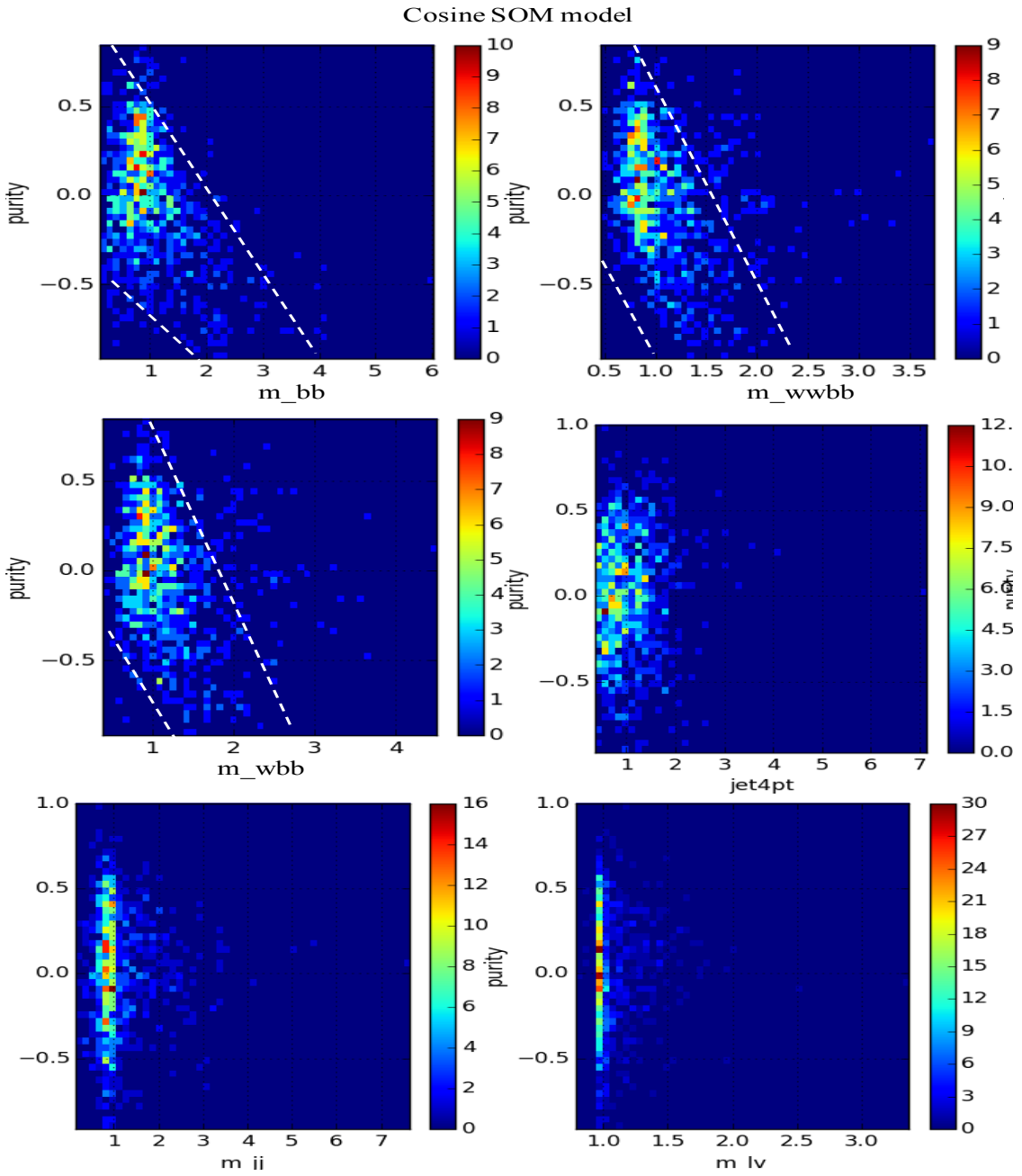


Figure 6.11: The 2-Dimensional histogram of centroid purity versus certain feature magnitude for the Cosine SOM model

From this result, it can be concluded that if the signal were to be separate from the background using a simple form of "cuts". The separation should be done by using the lepton pT, m_{wbb}, MET, m_{wwbb} and m_{bb} as the parameters and the threshold values be around 1.

6.4 SOM +LDA/QDA Model Conclusion

In many other investigation, ANN, BDT, RF, and SVM scored higher than the LDA in various classification metrics (accuracy, sensitivity, AUC, etc.). However, the result from an LDA model is easier to interpret than other ML algorithms. The research carried out in this chapter proves that creating a stacking classification model with an SOM model such as the SOM+LDA improve the result. The improvement was shown to be due the capability of SOM to create local instance clusters (LIC) with high purity.

However, the stacked model of SOM+QDA did not produce any worthwhile improvement in comparison to the QDA model alone. The correlations between the centroid LIC purity and the centroid weight-vector feature revealed that Lepton Pt, m_{wbb}, MET, m_{wwbb}, and m_{bb} provide the best signal-noise discriminations.

CHAPTER 7
SOM FOR CLUSTERING

"Ignorant readers are apt to judge a writer by his reputation. For my part, I read only to please myself. I like nothing but what makes for my purpose."

(Voltaire, 1759, p.84)

7.1 Chapter Introduction

In this chapter, the application of SOM in finding a hidden pattern in a particle physics dataset is demonstrated. The first half of the chapter depicts the clustering of the SUSY dataset instance using a stacked model named SOM+DPGMM. The SUSY dataset had been chosen since the difference between signal and noise in this dataset is more distinct than in the Higgs dataset. The signal and noise instances in the Higgs dataset too similar, requiring a large size SOM map to fully extract the hidden pattern.

The second half of this chapter demonstrates the usage of SOM on a real CMS result dataset, the dimuon dataset. The SOM was used to uncover the hiddent resonances signal that were hidden in the invariant-mass spectrum by the Drell-Yang process.

The details of the development pertaining to the application used to create the SOM model is given in Chapter 4, while the explanation regarding the hyperparameter configuration used to create the model is given in Chapter 5. The main hardware used to generate all the ML models in this chapter was done on the UM sifir cluster, previously

discussed in Chapter 3. The physics behind the SUSY and the Dimuon datasets is provided in Chapter 2.

7.2 SOM+DPGMM Model

The SUSY dataset was first used to create the required SOM model. The model has a map size of 30 × 30 centroid, thus producing 900 local instance cluster (LIC). Analyzing each of the 900 LIC one by one would be inefficient.

Therefore, the DPGMM algorithm was used to cluster the trained SOM centroids based solely on their weight-vector and then forming several classes of centroid. By doing this, the DPGMM also indirectly clustered the instances that were mapped to a given centroid. For example, instances Xa –Xc and Xd – Xf in **Figure 7.1** were mapped to centroids 1 and 2 respectively, while the DPGMM algorithm clustered both centroids 1 and 2 in the same centroid class. Hence, instances Xa –Xc and Xd – Xf belonged to the same instance class. In this thesis, the class generated by the DPGMM is termed as 'DPGMM-class', thus instances Xa –Xc and Xd – Xf belonged to the same DPGMM-class.

Therefore, in this method, the instances were clustered by the SOM model centroid, while the DPGMM clustered the centroid of the SOM model. The DPGMM implementation was taken from the Scikit-learn python module, described by Pedregosa et al., (2012).

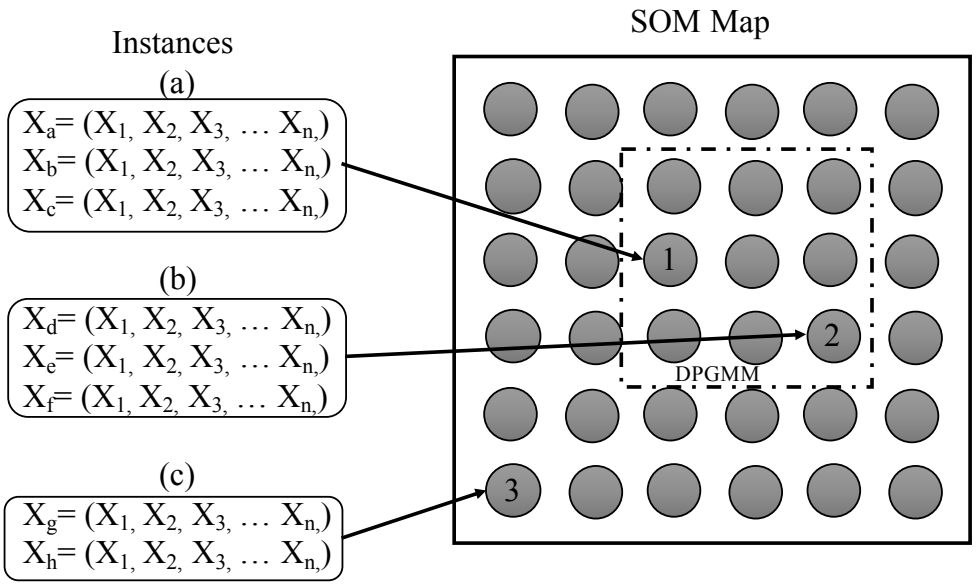


Figure 7.1: DPGMM cluster for both centroids 1 and 2 in the same cluster, thus all inherited instances (X_a –X_c and X_d – X_f) belonged to the same instance cluster.

7.3 SOM Algorithm Sanity Check

Before using the developed SOM algorithm (described in Chapter 4) on the SUSY dataset, the algorithm had to be validated to ensure it possessed the capability to cluster simpler datasets. The 'sanity-check' was done by using a dataset that contained 900 instances of points originated from 3 different 'blobs' distributed in a 3-Dimensional Euclidean space, as portrayed in **Figure 7.2**.

In the original dataset, each instance was labelled according to which blob-cluster it belonged to. However, the label information was dropped from the dataset when it was used to train the SOM model, thus the SOM algorithm did not have any information regarding instance labels (unsupervised learning). The hyperparameters that were used for the SOM model training are given in **Table 7.1**.

Once the training phase had been completed, the dataset instances were mapped back to the SOM map and they were re-labelled with their original label. The distribution of instances and their blob labels on the SOM feature-map is shown in **Figure 7.3**;

Table 7.1: The hyperparameter for the sanity check training

Hyperparameter	**Value or type**
Size	30 X 30 centroid
Shape	Square
Learning-Rate function	Derivative hyperbolic tan
Similarity function	Euclidean distance
Training iteration	2700

Figure 7.3 shows that the instances that belonged to the same blob tended to form clusters on the same area on the SOM feature-map albeit class 0 and 2 are defragmented. The results obtained here proved that the developed code was capable of forming correct clusters even without knowing the original instance label.

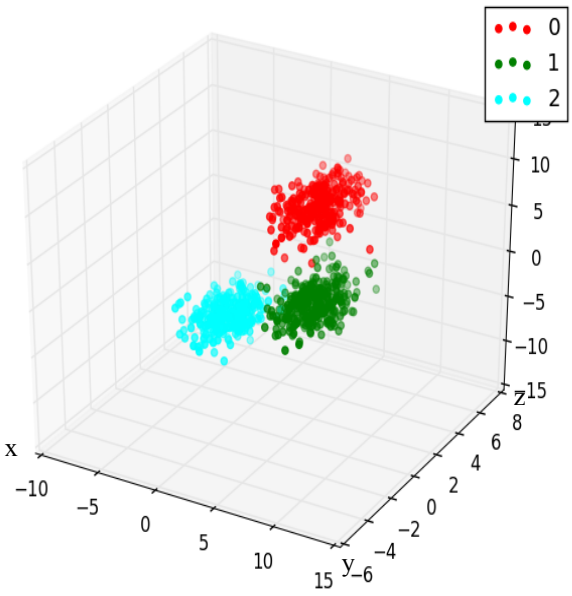


Figure 7.2: Scattering of points for the sanity-check dataset, which contained 900 instances of point.

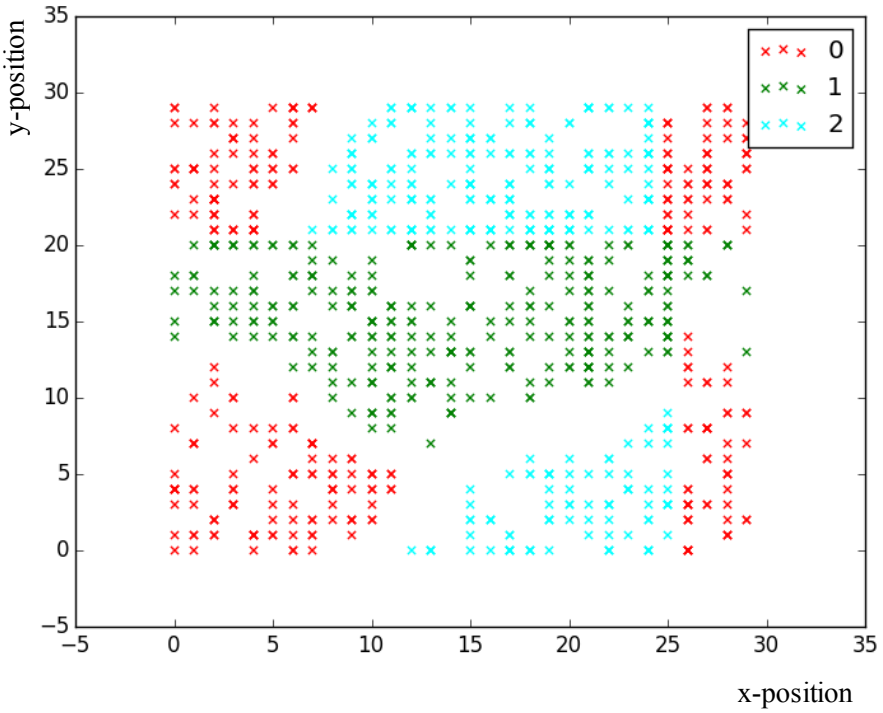


Figure 7.3: The distribution of instances on the SOM feature-map, which are colour coded dependent on their original blob cluster in **Figure 7.2**

7.4 Supersymmetry Dataset Clustering

The objective of this study was to test the capability of the SOM+DPGMM model to cluster the supersymmetry instance into clusters that consisted purely of noise or signal instances.

The SUSY dataset was clustered by using the SOM+DPGMM clustering model previously explained in subchapter 7.2. The dataset contained 500,000 instances, with signal-noise ratio of 0.5, however, the label of each instance in the dataset was removed during SOM modelling, mapping, and DPGMM clustering. As for the SOM model, it was trained by using the hyperparameters shown in **Table 7.2.** The reason for this configuration was discussed previously in subchapter 5.3.5.

Once the SOM model had been trained, centroids were clustered by the DPGMM algorithm, creating several DPGMM classes. The results from the DPGMM clustering are given in the next subchapter.

Table 7.2: The hyperparameter for the SUSY SOM

Hyperparameter	**Value or type**
Size	30 X 30 centroid
Shape	Square
Learning-Rate function	Hetero- Reverse Logistic Regression
Similarity function	Various
Training iteration	15 000

7.4.1 Clustering Results

The spectrum of magnitude for each centroid's weight-vector in the SOM model is shown in **Figures 7.4 (a-c)**, for Euclidean, City-block, and Chebyshev similarity functions. Meanwhile, Figure **7.4 (d)** shows the distribution of angle between each centroid weigh-vector and the mean weight-vector for the cosine SOM model.

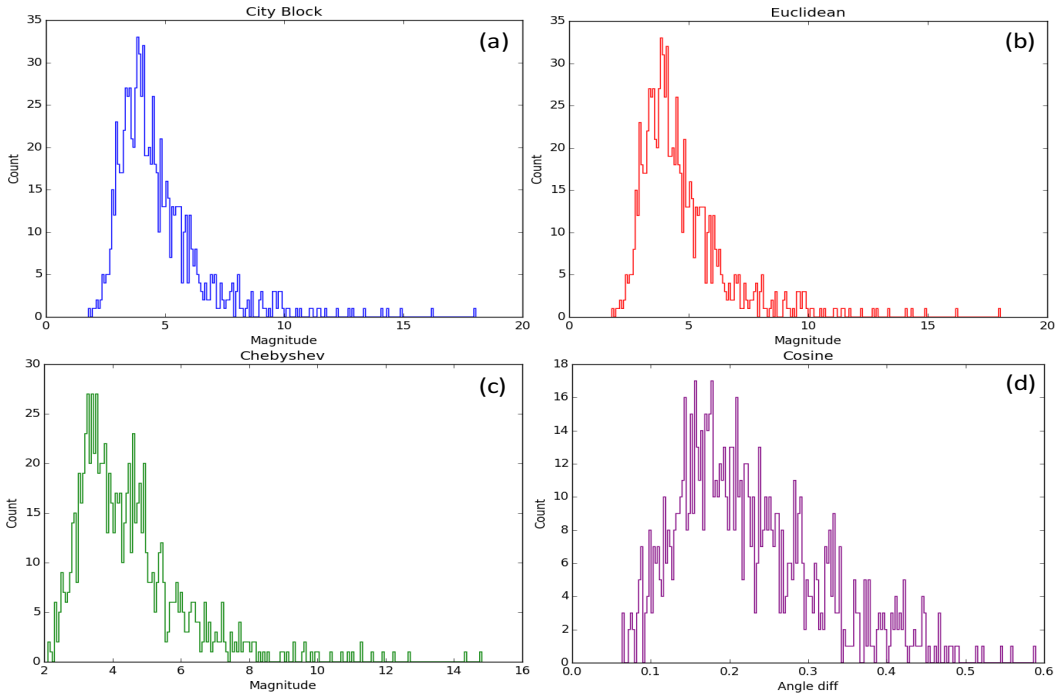


Figure 7.4: The distribution of centroid weight-vector magnitudes for Euclidean (a), city-block (b), and Chebyshev (c) SOM models, as well as the angle distribution for the Cosine SOM model (d).

Each distribution in **Figure 7.4** has a strong resemblance to the Gaussian mixture distribution; thus, it had been logical to cluster each distribution by using the Dirichlet Gaussian Mixture Model (DPGMM).

Using the DPGMM algorithm, the centroids in the SOM model were grouped into several classes based on their weight-vector. **Figure 7.5** shows the redistribution of the

centroid weigh-vector (for Euclidean, city-block, and Chebyshev models) and the angle (for cosine) for each DPGMM-class, whereas **Figure 7.6** shows the label of each centroid on a given SOM map based on their DPGMM-class.

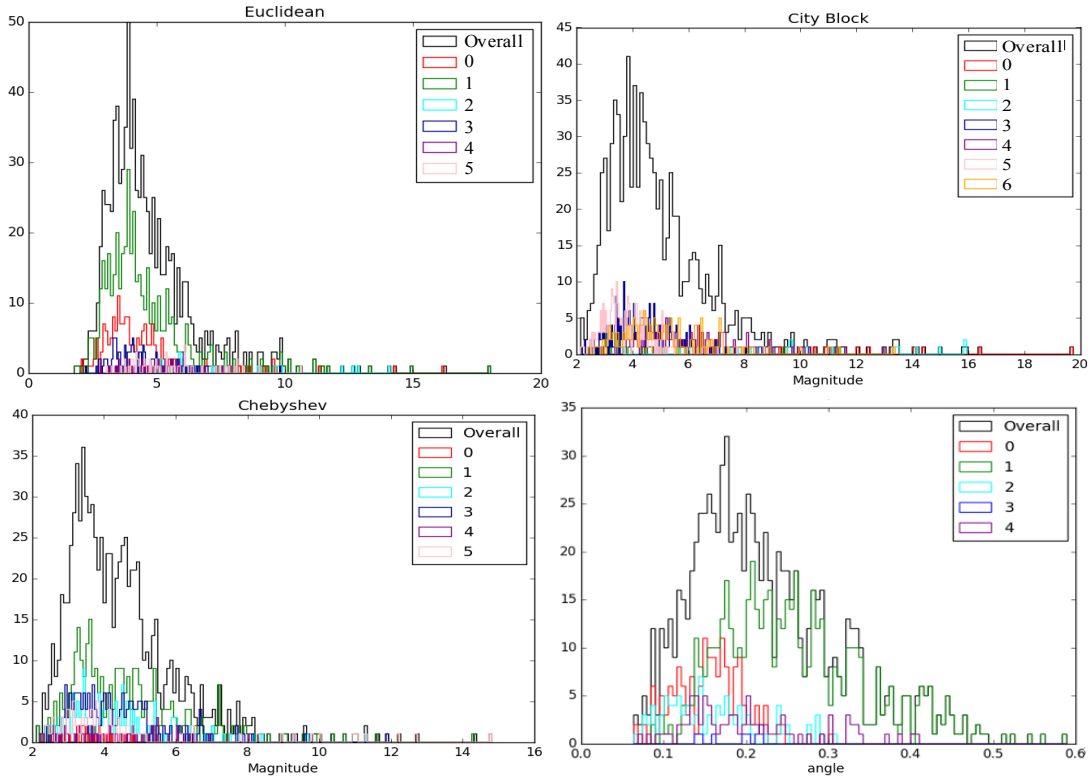


Figure 7.5: The distribution of centroid vector magnitudes (angle for cosine SOM) based on the group created by the DPGMM algorithm.

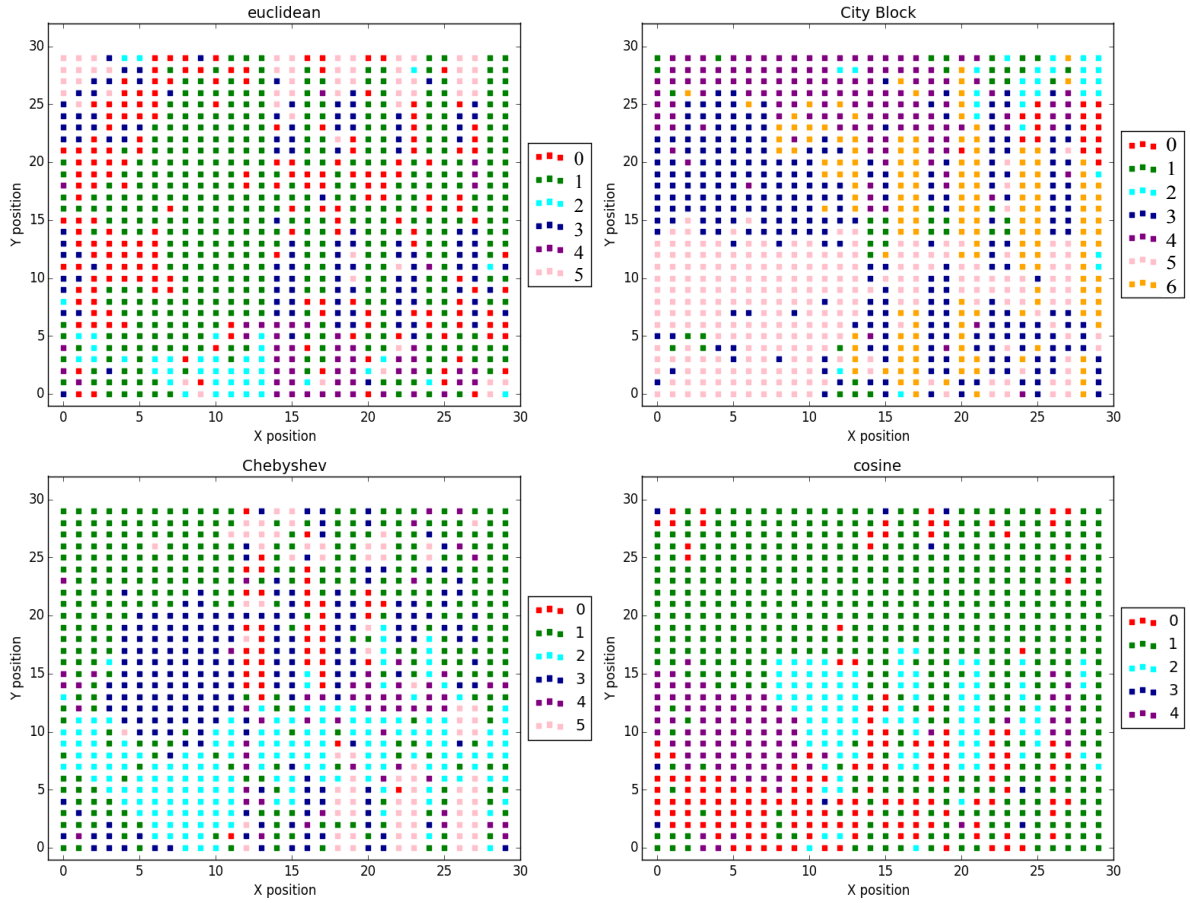


Figure 7.6: Centroid labels based on their DPGMM cluster for different SOM models

7.4.2 DPGMM class Purity Results

As stated before, the objective of this research was to study the capability of the SOM+DPGMM algorithm in developing a cluster of pure signal or noise instances. The instance purity of each DPGMM-class, P_{DPGMM}, was calculated based on equation (7.1), as shown in **Table 7.3**. A DPGMM cluster is said to be pure of signal if its $P_{DPGMM} = 1$, have an equal mix of signal and noise instances if the $P_{DPGMM} = 0$, and purely noise instance if the $P_{DPGMM} = -1$.

$$P_{DPGMM} = \frac{(Signal\ Instance\ Count - Noise\ Instance\ Count\ )}{Total\ Instance\ Count} \quad (7.1)$$

Table 7.3: The purity of each class generated by the DPGMM algorithm for various SOM models trained by using different similarity functions

Class	Instance Count	Purity	Class	Instance Count	Purity
Euclidean			City-block		
0	105954	-0.065	0	2619	0.521
1	275361	-0.199	1	13714	0.994
2	15815	0.513	2	4798	0.980
3	68218	0.298	3	184371	-0.137
4	17257	0.995	4	57008	0.610
5	17395	0.928	5	185271	-0.290
	Chebyshev		6	52219	0.469
0	15994	0.996		Cosine	
1	202336	-0.133	0	92478	0.267
2	102403	-0.188	1	331237	-0.234
3	118942	0.193	2	41922	0.583
4	31118	-0.370	3	4514	-0.226
5	29207	0.648	4	29849	0.983

It was apparent when analysing **Table 7.3** that there is a relationship between the DPGMM class purity and the class instance count, as **Figure 7.7** shows that the purity of class tend to decrease as the class instance count increased. This relationship is similar the results in classifying Higgs instances by using SOM+LDA.

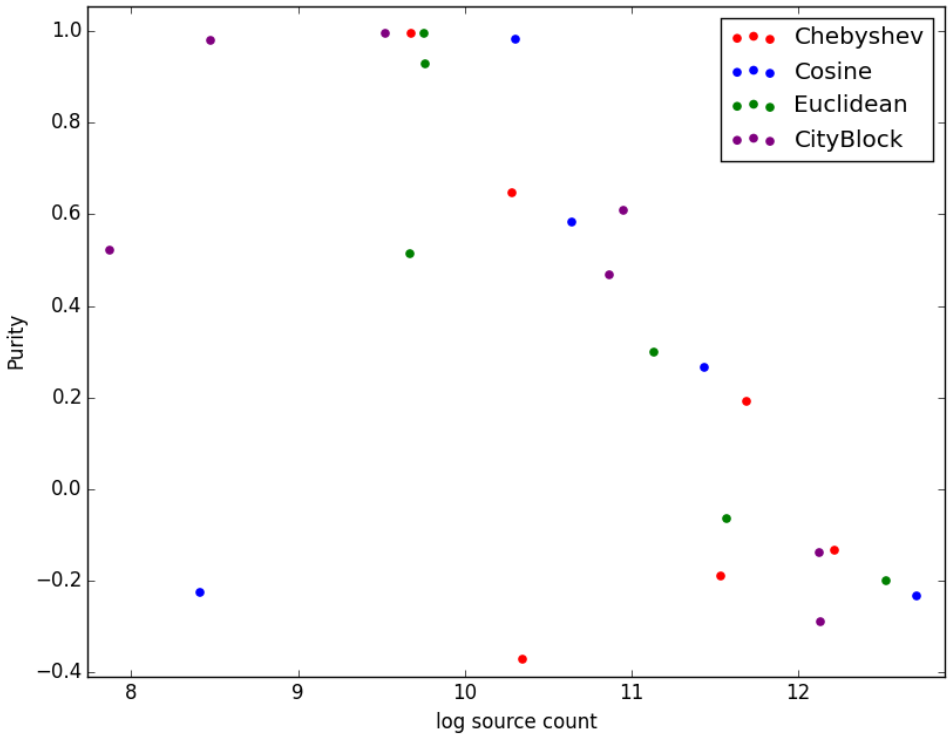


Figure 7.7: The purity of the class versus the number of instances

Before discussing the results obtain here, a more detailed discussion at instance mapping by the SOM is given.

7.4.3 SOM Model Instance Mapping Results

The two important parameters that should be reviewed after all instances have been mapped to the SOM model are the LIC count distribution and the LIC purity distribution across the SOM map. See subchapter 4.3.3 for an explanation of LIC and SOM mapping methods. **Table 7.4** gives a parametric review regarding the LIC count for each centroid on the SOM map, while **Figure 7.8** shows the LIC count distribution for all centroids across the SOM map.

Table 7.4: The mean, the standard deviation (STD), the min, and the max of centroid LIC count for SOM trained with various similarity functions.

Similarity Function	Centroid LIC count			
	Mean	STD	Min	Max
Chebyshev	555.556	258.423	67	1918
Euclidean	555.556	266.395	50	1848
Cosine	555.556	327.529	55	2038
City-block	555.556	356.330	12	2386

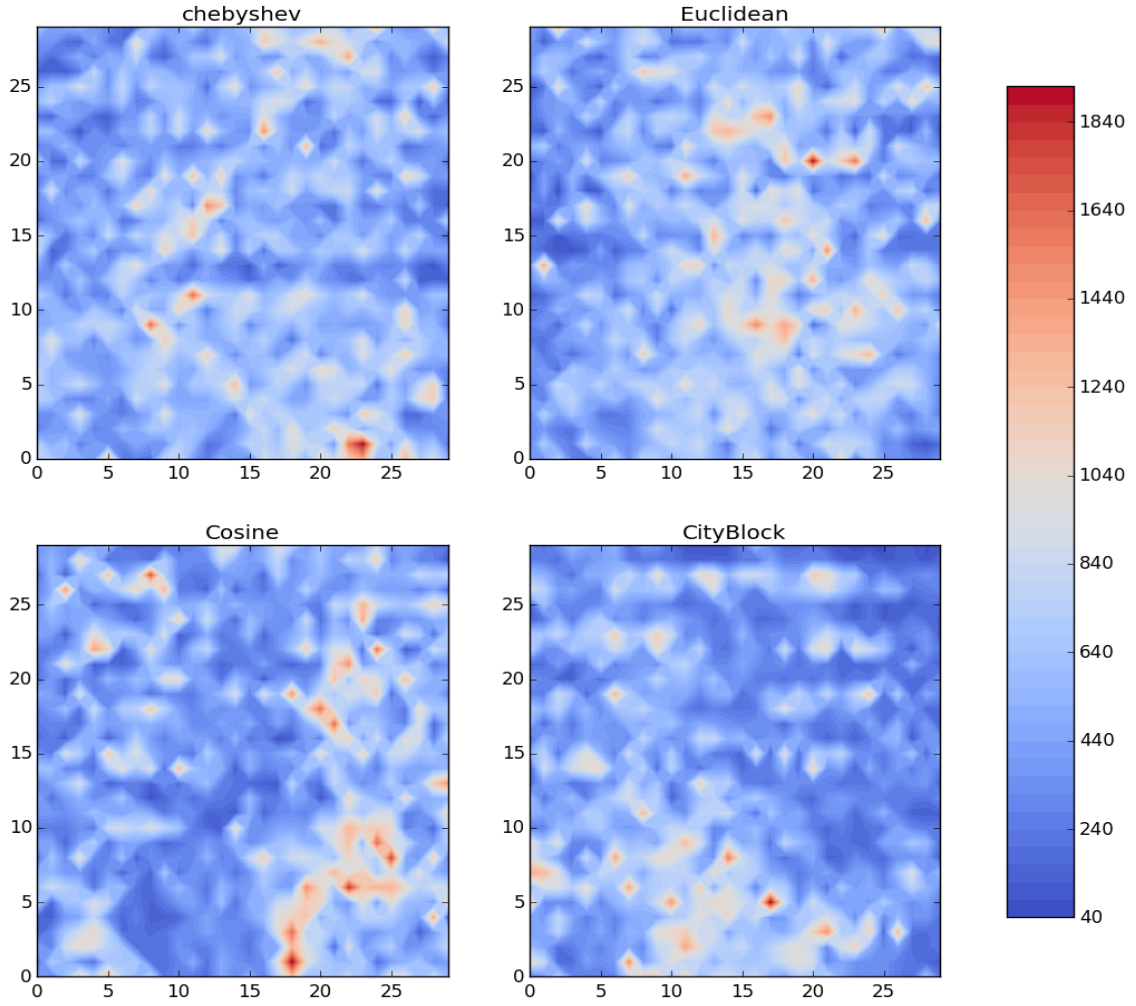


Figure 7.8: The distribution of centroid LIC count across the SOM feature-map.

The centroid LIC purity was measured by using equation (6.5). **Figure 7.9** illustrates the LIC purity distribution across the SOM map, and a parametric review is given in **Table 7.5**. μ_P, and $\|\mu_P\|$ refer to the mean purity of all centroids and the absolute

purity of all centroids for a given SOM model, respectively. **Table 7.5** also shows the percentage of centroid for a given SOM model with a purity value, *P,* exceeding 0.0, 0.5, and 0.8. Colour is used in table to indicate which box has the highest or lowest value for a given purity percentage. In addition, **Figures 7.10 – 7.13** are 2-dimensional histograms of centroid LIC purity versus centroid LIC count. The colour in the histogram show the number of centroid that has a given LIC count and LIC purity.

Table 7.5: The mean, μ_P , and the absolute mean, $\|\mu_P\|$, for centroid purity for each SOM model, as well as the percentage of centroids with purity exceeding 0.0, 0.5, and 0.8

Similarity Function	μ_P	$\|\mu_P\|$	% centroid		
			$P > 0.0$	$P > 0.5$	$P > 0.8$
Euclidean	0.139	0.524	0.523	0.342	0.233
City-block	0.256	0.565	0.607	0.441	0.276
Chebyshev	0.062	0.463	0.451	0.264	0.166
Cosine	0.195	0.525	0.572	0.381	0.231

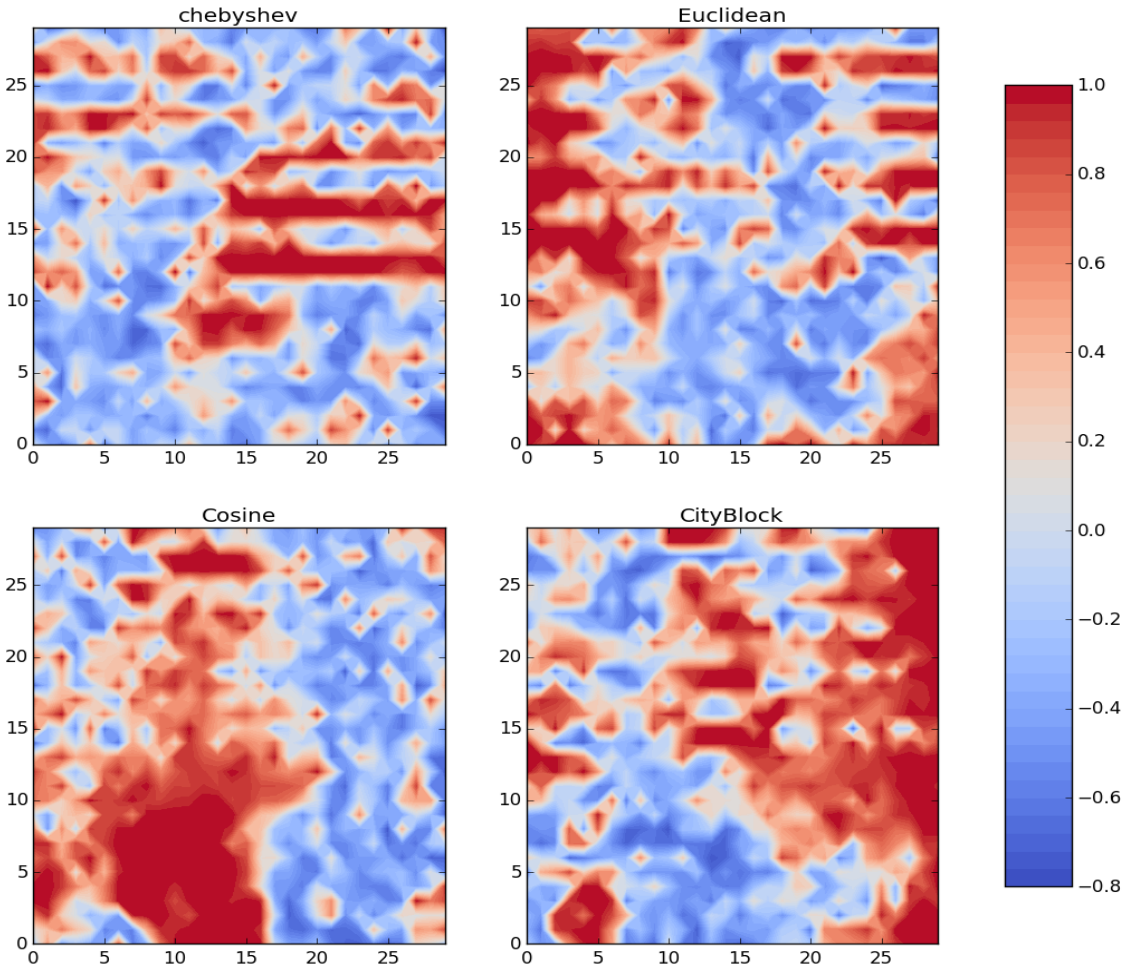


Figure 7.9: The purity of each centroid on the SOM feature-map for different SOM models.

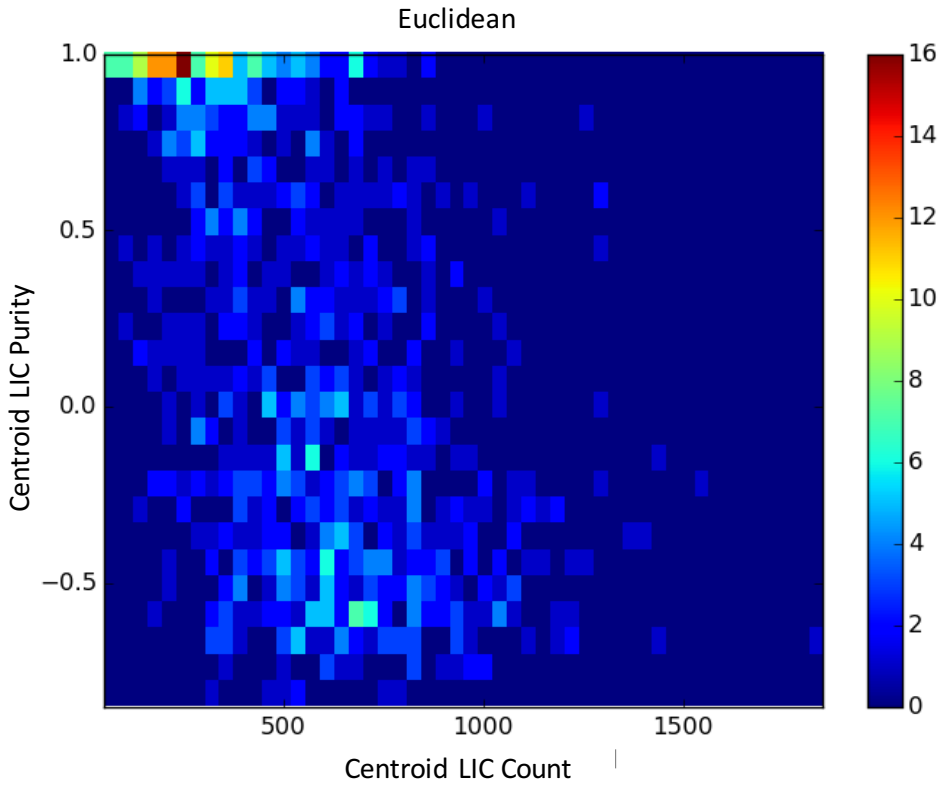


Figure 7.10: Centroid LIC purity vs LIC count for the Euclidean SOM

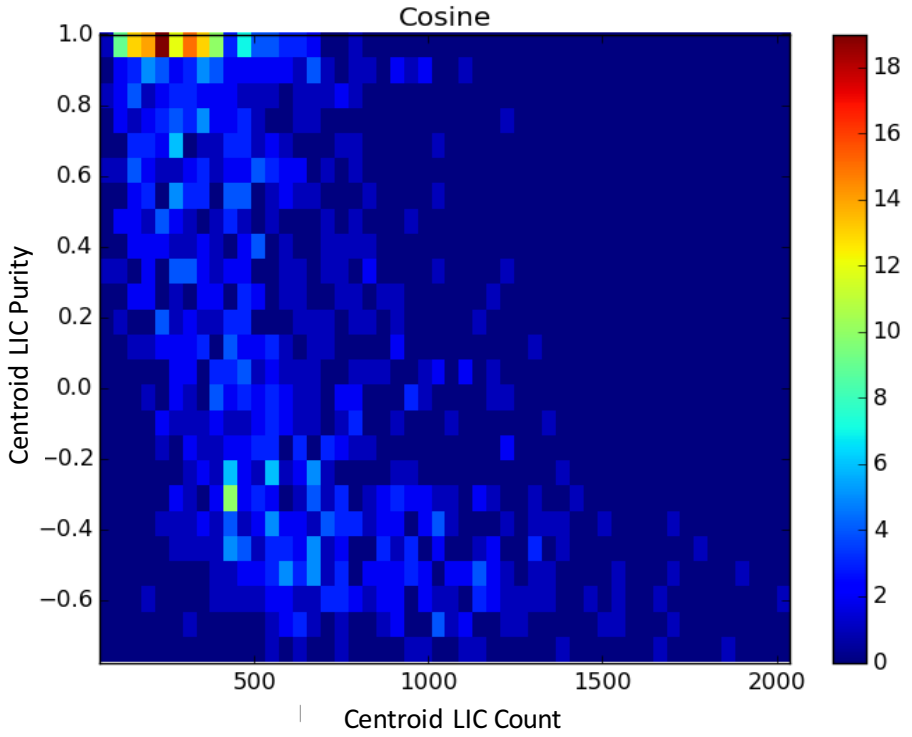


Figure 7.11: Centroid LIC purity vs LIC count for the Cosine SOM

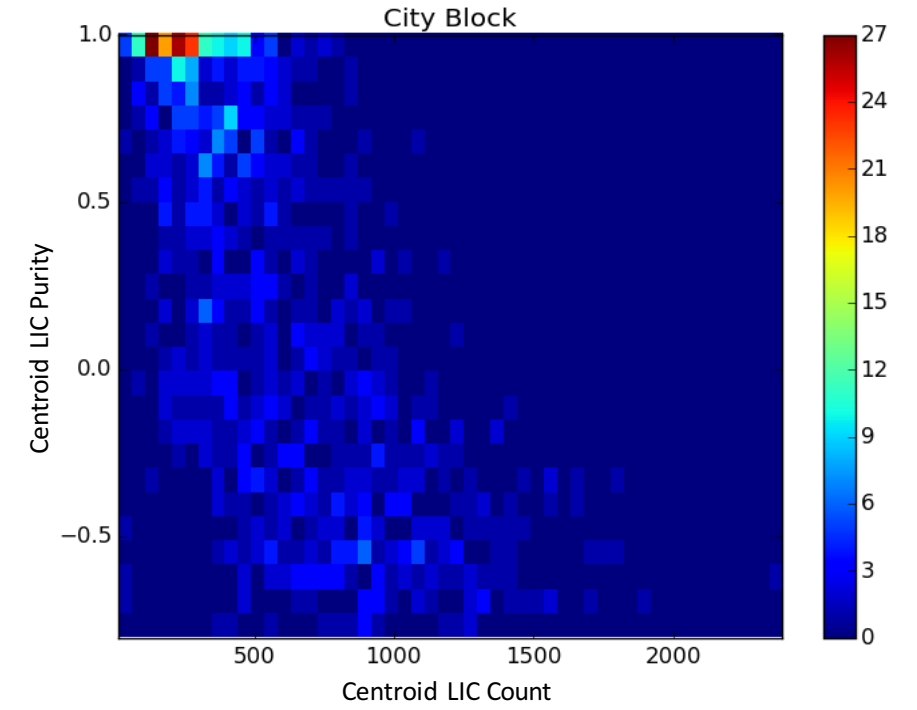


Figure 7.12: Centroid LIC purity vs LIC count for the City-block SOM

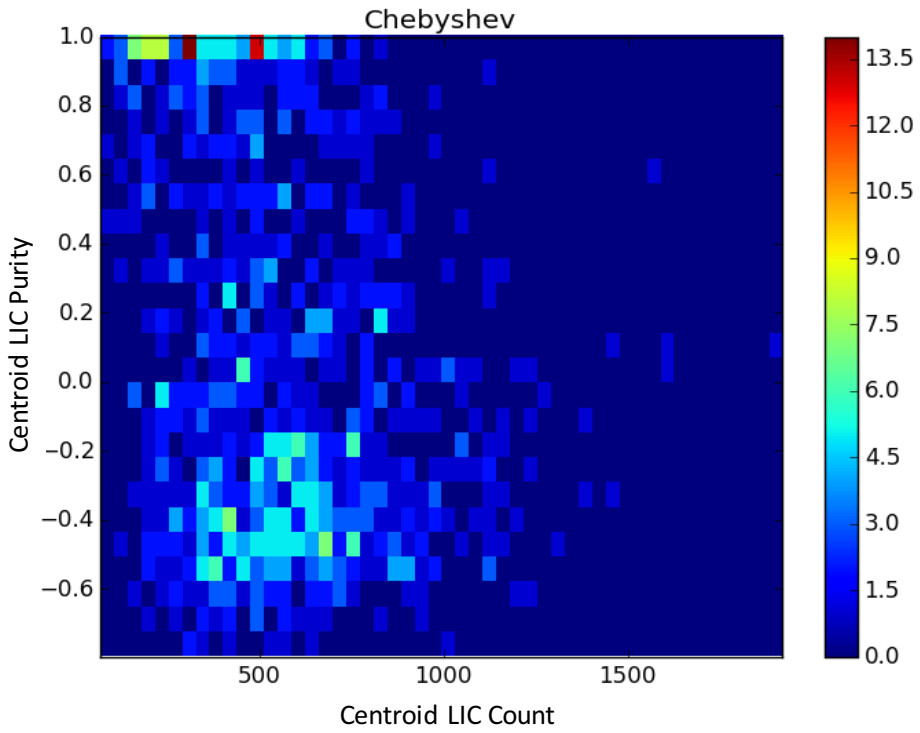


Figure 7.13: Centroid LIC purity vs LIC count for the Chebyshev SOM

7.4.4 SOM+DPGMM Clustering Discussion

The objective of this research was to study the capability of SOM+DPGMM model to cluster the SUSY dataset into clusters of pure signal and noise. The clustering was done in an unsupervised manner, in which the algorithm did not have any information regarding the original label (signal/noise) of each instance and had to cluster instance solely based on the value of the feature parameters.

The results portrayed in **Table 7.3** indicate that each SOM+DPGMM model was capable of producing a DPGMM class that contained purely signal instances, but failed to generate any class purely of noise instances. The Euclidean model created the DPGMM class (combination) with the highest signal instance, followed in order by the cosine, city-block, and Chebyshev models. The results obtained here further substantiate the claim made by Kohonen (2013) that the Euclidean distance function is the most suitable similarity function for the SOM model.

The DPGMM algorithm created classes based on the centroid weight-vector, but information regarding the content of the centroid ILC was not given. Thus, any DPGMM class purity is determined by the ILC purity of its affiliated centroid. Hence, the purity of any DPGMM class is determined by the outcome of the SOM modelling, and not the DPGMM algorithm. **Figures 7.10 - 7.13** show no centroid created by any of the SOM models had high noise purity (LIC purity < -0.8). Thus, it would be unlikely for the DPGMM algorithm to produce a class that contains purely noise instances as verified **Table 7.3.**

It is interesting to note that since **Figures 7.10 - 7.13** displayed no centroid with ILC purity < -0.8, signal instances had been spread throughout the SOM map, whereas noise instances were focused in a particular region in the SOM map. This is verified in **Figure 7.14**, where the noise instances tend to cluster at the middle area section of the SOM map, while the signal instances were spread out across the SOM map. This indicates that the noise instance feature values had less variance compared to the signal instances.

The noise instances are derived from a particle physics event with 4 final products ($l^+l^-\upsilon\upsilon$) and only two undetectable particles, whereas the signal instance derived from an event that had six final products ($l^+l^-\upsilon\upsilon\chi^0\chi^0$) and four undectable particles. Thus, the signal instance had a higher number of final particles and hence a broader spectrum (high variance) of missing energy compared to the noise instance. This explains why the noise instance feature value exhibited less variance compared to the signal instances.

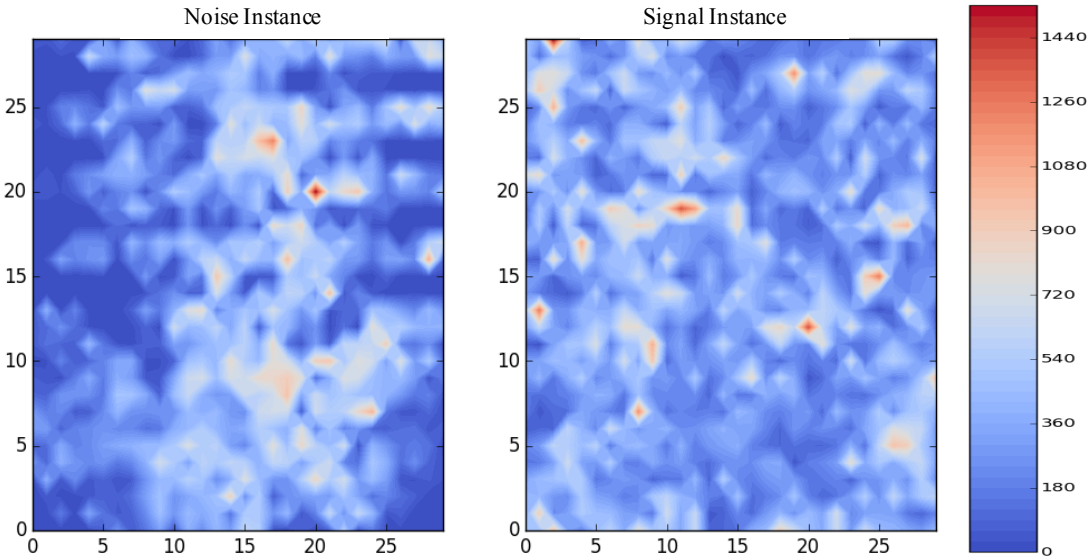


Figure 7.14: Comparison between the signal and the noise instance distributions on the Euclidean SOM map.

The missing momentum and M_R^T (defined by equation 2.7) are the two important features that differentiated the signal from noise in the SUSY dataset. The spectra of these two features for the DPGMM class with purity, $P > 0.95$, for each SOM model is given in **Figures 7.15** and **7.16**. In both figures, the Euclidean model had almost similar distribution with the real signal distribution for both M_R^T and missing momentum features. This shows that the Euclidean distance had been the most suitable similarity function in creating a SUSY SOM.

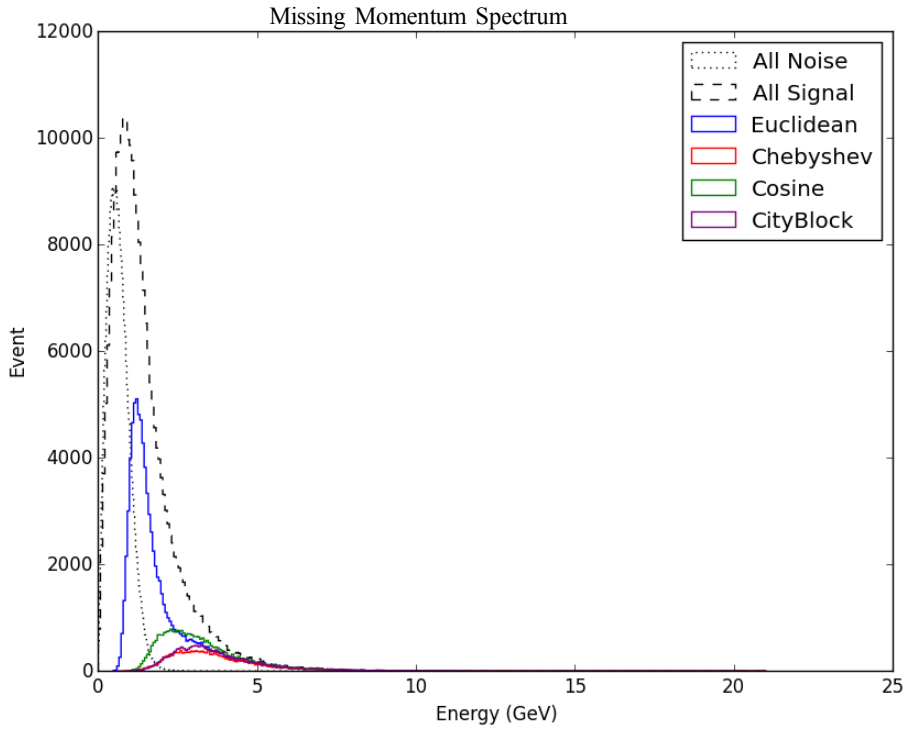


Figure 7.15: The missing momentum spectrum for signal dominant DPGMM-class

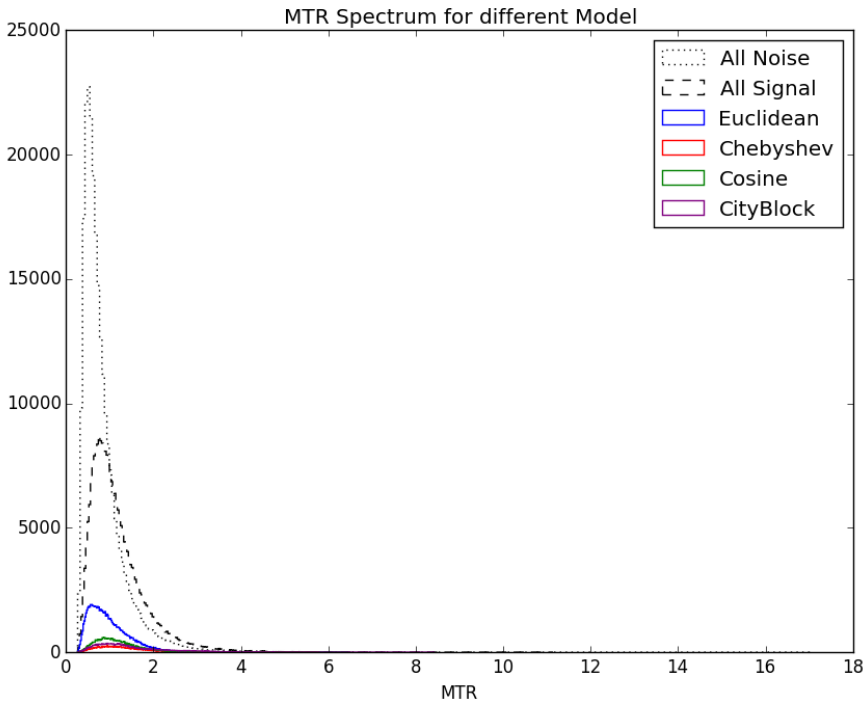


Figure 7.16: The MTR spectrum for signal dominant DPGMM-classes

7.4.5 Significance

Several clustering algorithms have been employed in particle physics analysis, including the kt and anti-kt jet clustering; however, these algorithms are particle physics domain specific and cannot be used outside particle physics analysis. The result obtained in **Table 7.3** clearly show that the SOM can create a class of pure signal, in the case of the city-block SOM model, a purity of 0.994 was recorded. This result is the most important in the thesis as it demonstrates that a non-particle physics algorithm can be used as a clustering tool in particle physics.

Another point that the author would like to point out is the usage of classification algorithm versus clustering algorithm. Several examples of classification algorithm usage have been given in Chapter 6, however, a classification algorithm is designed to only classify instances to a predefined class or label. This means; a classification algorithm

would classify a novel data (or an abnormal data) into a predefined class, instead of creating a new class as a clustering algorithm would. An algorithm that can automatically generate a new class for abnormal data can avoid the mistake of accidentally removing abnormal data that otherwise may lead to a significant scientific discovery.

7.5 Dimuon Clustering

Several cases before this showed that centroids with low LIC number store more valuable information than an LIC with high number. For example, in the SOM+LDA classifier model, centroids with low LIC count were seen to have e a higher purity and increased the overall ROC-AUC score. For the SOM+DPGMM case, **Figure 7.17** shows that centroids with LIC purity > 0.8 tend to have a LIC count of fewer than 500, while LIC with lower purity (purity < 0.8) had a mean count of 616.849 and 615.893 for Euclidean and Cosine SOM models, respectively. Both these examples seem to imply that, for SOM model, valuable information or pattern tends to be stored inside centroids with a low LIC count.

The previous two datasets (SUSY and Higgs datasets) that had been used were simulated datasets. In the subsequent text, a dataset of real recorded particle events at the CMS detector has been used to train the SOM model. The dataset is the dimuon dataset which discussed in subchapter 2.2.2. The objective of the next subchapter is to study the possibility of finding a hidden pattern (with physical meaning) among SOM centroids with low LIC count.

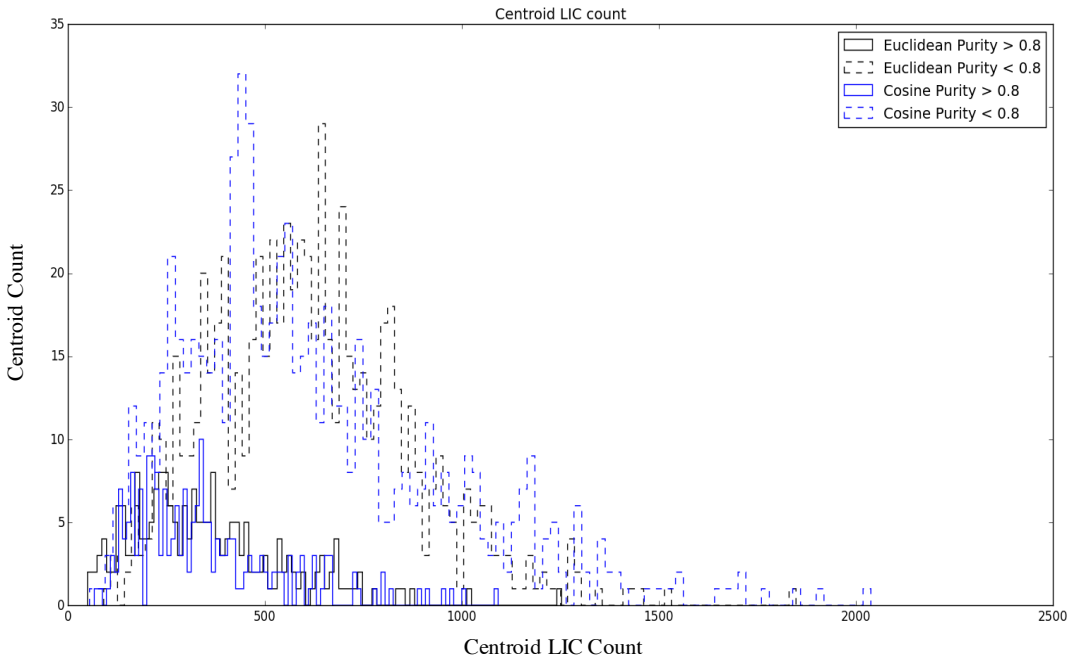


Figure 7.17: The comparison of centroid LIC count between centroid with LIC purity < 0.8 and > 0.8 for Cosine and Euclidean SOM models.

7.5.1 Dimuon SOM model

The dimuon SOM model was trained by using the hyperparameter configuration stated in **Table 7.6**. The momentum on x, y, and z-axes and the energy for both muons was chosen as the features to be used to generate the SOM model. The invariant mass of the dimuon was excluded in the training feature, thus, the SOM model that had been developed did not have any direct information regarding the dimuon invariant mass.

Table 7.6: The hyperparameter for to train the SOM model on the dimuon dataset.

Hyperparameter	Value or type
Size	100 X 100 centroid
Shape	Square
Learning-Rate function	Homo- derivative hyperbolic tan
Similarity function	Cosine
Training iteration	10,000

Once the training had been done, the dataset was mapped back to the SOM model. The log of LIC count for each centroid on the SOM map is given in **Figure 7.18** centroids with LIC count = 0 are given the value of -1.

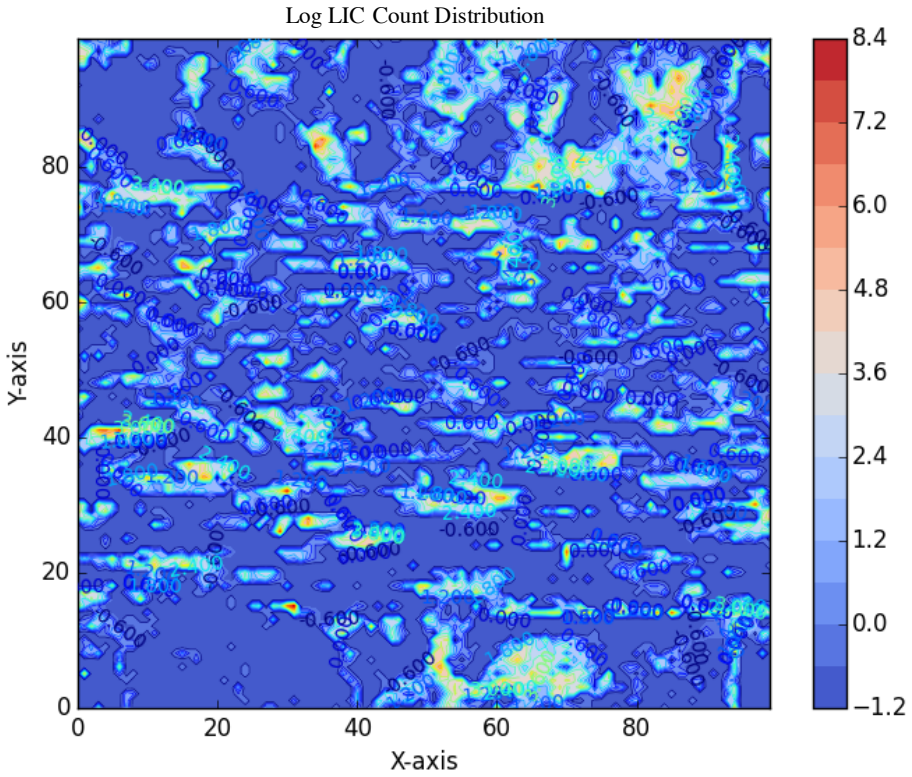


Figure 7.18: The log LIC count distribution on the SOM map for the dimuon dataset. Centroid with LIC count = 0, is given the value -1.

7.5.2 Dimuon Invariant Mass

Figure 7.19 show the mean of the invariant mass in log scale for each instance in a centroid LIC, with centroid with LIC count = 0 given the value -1 in the spectrum. **Figure 7.20** show the distribution of the same parameter (mean of the log-invariant mass) for each centroid across the dimuon SOM map.

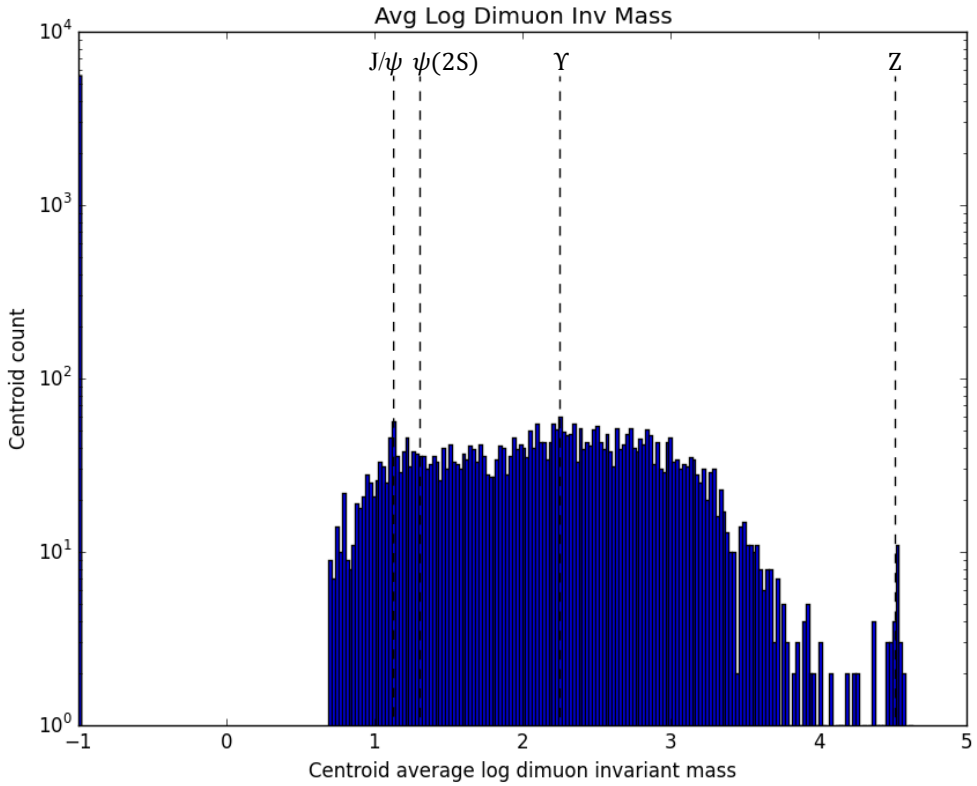


Figure 7.19: The spectrum of the centroid LIC average log-invariant dimuon mass, centroid with LIC = 0, was given the value -1.

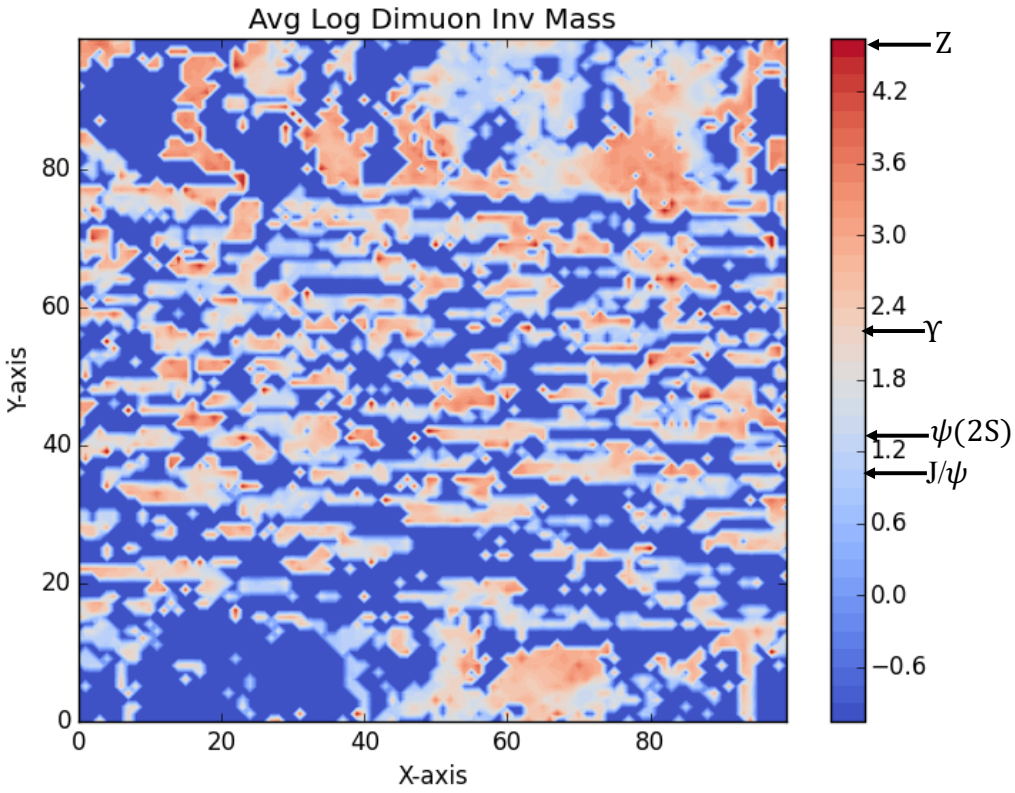


Figure 7.20: The average of log-invariant mass of dimuon for each centroid on the SOM map.

7.5.3 Low Count LIC Centroid

Figure 7.21 shows the 2-dimensional histogram of the log-LIC count versus the log of the average invariant mass, (centroids with LIC count = 0 neglected) for all centroid in the dimuon SOM model. In this figure, a pattern emerged among centroids with log LIC count < 1.39 (LIC count < 4). In this region, there was a sudden increase in the number of centroids (relatively more red to its adjacent bin) when the average of log-invariant mass was equal to a given resonance mass.

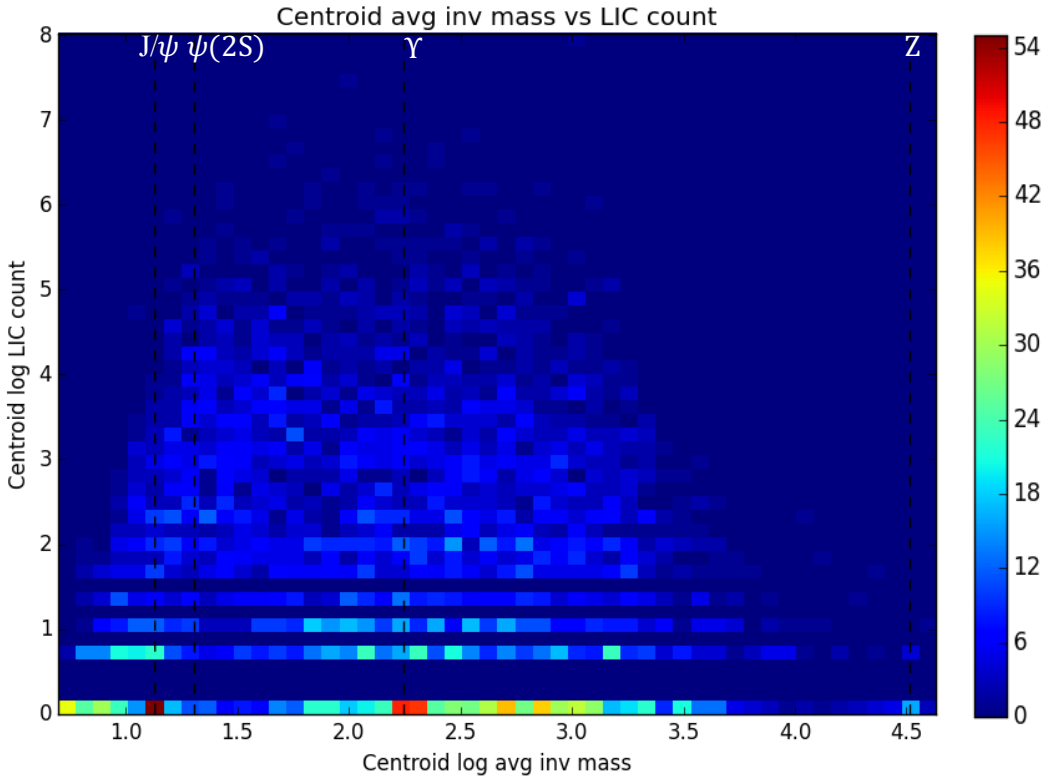


Figure 7.21: 2D-Histogram of centroid log LIC count versus the centroid log average invariant mass.

The invariant mass of all instances from LIC that has instance-count less than 4 is shown in **Figure 7.22**. From this histogram, the peak from J/ψ and $\Upsilon(1s)$ can clearly been seen above the background. Only the $\psi(2S)$. The higher number of centroids having average log invariant mass at a resonance invariant mass is shown more clearly in **Figure 7.22**, where all LIC that had count < 4 had been plotted in a histogram the LIC average log invariant mass. This figure clearly shows that several centroids peaked about its neighbouring bin when the average invariant mass had been equal to a resonance mass. Moreover, **Figure 7.22** displays the three peaks for the three upsilon resonance, which were missing in **Figure 7.19**. Only the resonance of $\psi(2S)$ is not clearly visible in **Figure 7.22**.

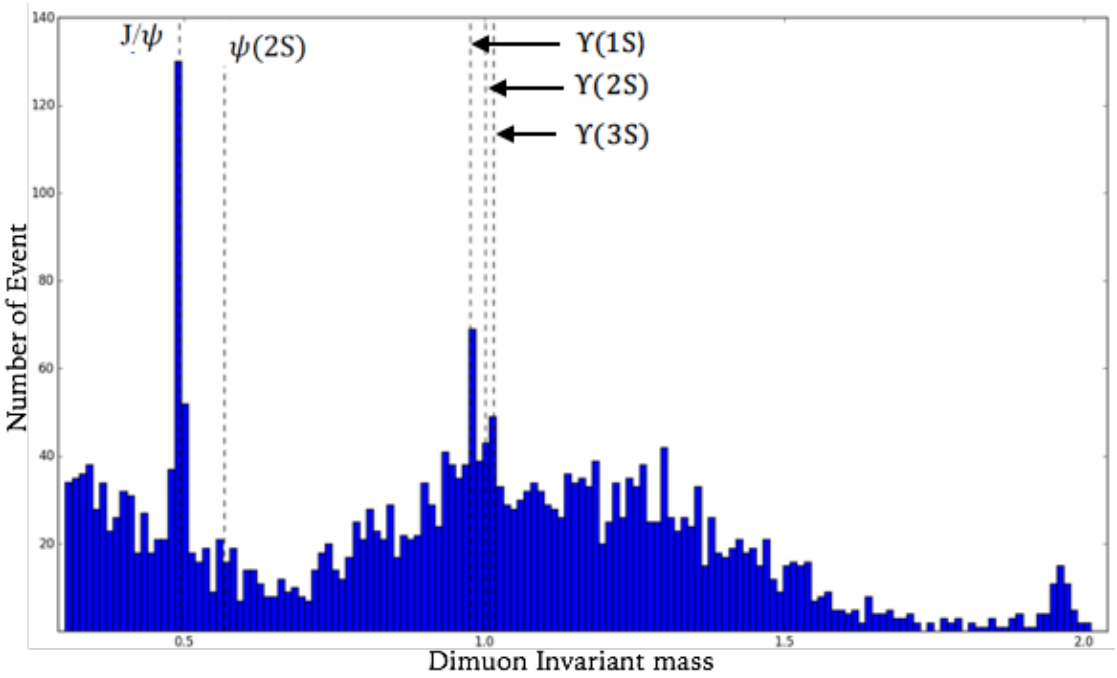


Figure 7.22: The number of centroids for a given log average invariant mass, for centroid in the SOM model with LIC count < 4

The significance of **Figure 7.22** is that it shows that the SOM was able to cluster the large bulk of noise instances (produced by the Drell-Yang process) to only certain centroids, which were subsequently discarded in the analysis for having high LIC count (>= 4). Meanwhile, the centroid with low LIC, (< 4), count produced a pattern of invariant mass distribution with a physical meaning. Another significant attribute of this figure is that the resonance peaks had been more apparent in comparison to those illustrated in the original invariant mass spectrum in **Figure 2.6.**

7.6 SOM Clustering Conclusion

The results of the study derived from this chapter demonstrate that the SOM+DPGMM model could generate cluster(s) with high signal purity level for the SUSY dataset in an unsupervised learning manner. It stands as an example that a non-particle physics-domain specific clustering method can still be used for particle physics

study. It was also found that the SOM model with Euclidean similarity function offered the best separation between signal and noise instances for the SUSY dataset.

On another hand, the results obtained from the dimuon SOM model showed that the SOM algorithm could extract hidden pattern from an experimental dataset. In both SOM models (SUSY and Dimuon SOM models), valuable information had been retrieved from centroids with low LIC count. Moreover, the SOM output from both datasets showed that noise displayed a tendency to cluster together, as well as to generate clusters with a high number of instances.

CHAPTER 8
CONCLUSION

8.1 Conclusion

The work carried out in this thesis examined holistically various computational methods employed in the particle physics analysis. This began with the computational practice of the Compact Muon Selenoid (CMS) Collaboration that operates a global computational grid infrastructure. The CMS experiment uses various middleware to manage users and resources available under the grid infrastructure, including PhEDEx, DAS, WMAgent, and CRAB. The WMAgent and CRAB are the front ends for the production and the analysis workflow respectively, with both the workflow submitted to their computing jobs to the grid infrastructure through the glideinWMS global pool.

Other than providing means for users to submit their personal analyses to the grid, CRAB also shields the end-users from the technical complexity of the CMS grid. CRAB version 3 (CRAB3) was developed to replace as older version (CRAB2), and various efforts by the author to these application developments. The contributions include the improvement of CRAB3 error-reporting mechanism, validation of users' read/write permission, parallel remotecopy, and the CRAB3-Client API.

In addition, a significant development effort was also undertaken in establishing a new HPC for University Malaya called sifir, acting as a CMS Tier-2 site. Effort were made to make sifir a computing cluster-centric rather than grid-centric.

Another computational technique that is commonly employed in particle physics analysis is Machine Learning (ML). This term covers various algorithms, including Artificial Neural Network (ANN), Random Forest (RF), Support Vector Machine (SVM), Self-Organizing Map (SOM), Linear/Quadratic Discrimination Analysis (L/QDA), and K-Means. Since other implementations of SOM could not scale to the size and the complexity of particle physics datasets, a new SOM implementation was developed from scratch for this research. This implementation permitted various hyperparameters to be configured in order to adapt to the size and the complexity of a given dataset.

The effect of different hyperparameter configurations on the overall SOM centroid weight-vector has been studied; this included , training-iteration length, and learning-rate function. It was found that the best hyperparameter configuration for the SOM model designed to be applied to the Higgs data set was to have the logistic regression learning-rate on a homogenous mode as and the training iteration set to 7,500. Meanwhile, the SUSY SOM model was best configured to have hyperparameters that included the reverse logistic regression in heterogeneous mode as the learning-rate function and a training iteration of 15,000 steps.

Using the given hyperparameter configurations for the Higgs SOM model, two new classification models were proposed; the SOM+LDA and the SOM+QDA models. In the study conducted, the SOM+LDA displayed better classification capability on the Higgs dataset, in comparison to the LDA and the QDA models alone. The model also exhibited greater precision than SVM, but it did not perform better than the RF in any classification scoring parameter. The SOM model was found to enhance the classification performance by creating a sub-cluster that was purely or almost purely of signal or noise

instance. The SOM+QDA also failed to give a better performance compared to any other classification model in any scoring parameter.

This research also introduced the SOM+DPGMM model, which was used to cluster the instances from the SUSY dataset. In this model, the SUSY instance was first clustered by the SOM centroids to produce multiple LIC. After that, the DPGMM model clustered the SOM centroids, which indirectly clustered the LICs, to form different classes of instances. Through this method, one or two class(es) with signal purity exceeding 92% was successfully produced. The ability of the SOM+DPGMM model in producing class(es) with highly pure signal instance demonstrated that a non-physics-domain clustering algorithm indeed possessed the potential to cluster particle physics events.

The final research topic that was conducted was to study the SOM algorithm clustering capability on the dimuon dataset. It was found that by removing instances that were mapped to the centroid with high LIC count, sufficient noise was removed to reveal several particle resonance peaks in the invariant mass spectrum. Hence, this study establishes that the SOM model had the potential to uncover hidden patterns in a particle physics dataset.

8.2 Suggestions

One of the original objectives of this research was to study several methods that could be used as a novel detection technique in particle physics analysis. Such novel detection model presented an immense potential to provide an analysis that are physics-model-independent. However, due to lack of maturity in the novel detection algorithm,

as well as limited computing resource, such objective could not be achieved. The obvious choice when conducting the novel analysis is to use either the Mahalanobis distance statistics, nevertheless, the author suspected that coupling it with the SOM sub-clustering mechanism, a more powerful novel detection can be established.

The Mahalanobis distance is also another form of the similarity function. However, the function was not implemented in the development of the SOM model as the function is prone to singular matrix error in constructing the covariance matrix. Another function that can be exploited as a similarity function is the Kullback-Leibler divergence, however, this function is not a pair-wise function. Thus, further research has to be carried out to identify the type of modification that has to be done on the equation for it to be appropriate in the SOM modelling context.

Furthermore, in this research, inadequate study was made on the appropriate technique to leverage the GIC generated by the SOM map. In the past studies, the U-matrix had been employed to construct the boundary between the GIC, whereas in this research, the DPGMM algorithm was applied. Obviously, various methods are available to construct the GIC boundary, such as the Density-Based Spatial Clustering of Application with Noise (DBSCAN), as well as several hierarchical clustering algorithms. Thus, a closer look on this matter should be given consideration.

REFERENCE

Acat, P., & Heikkinen, A. (2007). Separation of Higgs boson signal from Drell-Yan background with self-organizing maps.

Adelman, J., Alderweireldt, S., Artieda, J., Bagliesi, G., Ballesteros, D., Bansal, S., … Zvada, M. (2014). CMS computing operations during run 1. *Journal of Physics: Conference Series*, *513*(3), 32040. http://doi.org/10.1088/1742-6596/513/3/032040

ALEPH Collaboration. (1990). ALEPH: A detector for electron-positron annihilations at LEP. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *294*(1–2), 121–178. http://doi.org/10.1016/0168-9002(90)91831-U

Alwall, J., Herquet, M., Maltoni, F., Mattelaer, O., & Stelzer, T. (2011). MadGraph 5: going beyond. *Journal of High Energy Physics*, *2011*(6), 128. http://doi.org/10.1007/JHEP06(2011)128

Amerijckx, C., Verleysen, M., Thissen, P., & Legat, J.-D. (1998). Image compression by self-organized Kohonen map. *IEEE Transactions on Neural Networks*, *9*(3), 503–507. http://doi.org/10.1109/72.668891

Andreeva, J., Belov, S., Berejnoj, A., Cirstoiu, C., Chen, Y., Chen, T., … Urbah, E. (2008). Dashboard for the LHC experiments. *Journal of Physics: Conference Series*, *119*(6), 62008. http://doi.org/10.1088/1742-6596/119/6/062008

ATLAS. (2012). Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Physics Letters, Section B: Nuclear, Elementary Particle and High-Energy Physics*, *716*(1), 1–29. http://doi.org/10.1016/j.physletb.2012.08.020

ATLAS Collaboration. (2012). Searches for supersymmetry with the ATLAS detector using final states with two leptons and missing transverse momentum in sqrt(s)=7 TeV proton–proton collisions. *Physics Letters B*, *709*(3), 137–157. http://doi.org/10.1016/j.physletb.2012.01.076

ATLAS Collaboration. (2013). Search for a multi-Higgs-boson cascade in WWbb events with the ATLAS detector in pp collisions at sqrt(s) = 8 TeV. *Physical Review D*, *89*, 52005. http://doi.org/10.1103/PhysRevD.89.032002

ATLAS Collaboration. (2014). Search for direct production of charginos, neutralinos and sleptons in final states with two leptons and missing transverse momentum in

pp collisions at sqrt(s) = 8 TeV with the ATLAS detector. *Journal of High Energy Physics*, *2014*(5), 71. http://doi.org/10.1007/JHEP05(2014)071

ATLAS Collabration. (2014a). A neural network clustering algorithm for the ATLAS silicon pixel detector. *Journal of Instrumentation*, *9*(9), P09009–P09009. http://doi.org/10.1088/1748-0221/9/09/P09009

ATLAS Collabration. (2014b). Search for the Standard Model Higgs boson decay to uu with the Atlas detector. *Physics Letters B*, *738*, 68–86. http://doi.org/10.1016/j.physletb.2014.09.008

BaBar Collaboration. (2002). The BABAR detector. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *479*(1), 1–116. http://doi.org/10.1016/S0168-9002(01)02012-5

Backer, E., & Jain, A. (1981). A Clustering Performance Measure Based on Fuzzy Set Decomposition. *Pattern Analysis and Machine Intelligence*, (1), 66–75. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4767051

Badala, A., Blanco, F., La Rocca, P., Pappalardo, G. S., Pulvirenti, A., & Riggi, F. (2008). Identification of the K*± resonance decay by topological cuts and multivariate discrimination methods. *The European Physical Journal C*, *56*(1), 17–26. http://doi.org/10.1140/epjc/s10052-008-0657-8

Bakhet, N., Khlopov, M. Y., & Hussein, T. (2015). Neural Networks Search for Charged Higgs Boson of Two Doublet Higgs Model at the Hadrons Colliders. Retrieved from http://arxiv.org/abs/1507.06547

Balcas, J., Belforte, S., Bockelman, B., Colling, D., Gutsche, O., Hufnagel, D., … Wissing, C. (2015). Using the glideinWMS System as a Common Resource Provisioning Layer in CMS. *Journal of Physics: Conference Series*, *664*(6), 62031. http://doi.org/10.1088/1742-6596/664/6/062031

Baldi, P., Sadowski, P., & Whiteson, D. (2014). Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, *5*, 4308. http://doi.org/10.1038/ncomms5308

Belforte, S., Gutsche, O., Letts, J., Majewski, K., McCrea, A., & Sfiligoi, I. (2014). Evolution of the pilot infrastructure of CMS: towards a single glideinWMS pool. *Journal of Physics: Conference Series*, *513*(3), 32041. http://doi.org/10.1088/1742-

6596/513/3/032041

Bellman, R. E. (1961). *Adaptive Control Processes: A Guided Tour* (Vol. 4). Princeton university press.

Bezdek, J. C., Ehrlich, R., & Full, W. (1984). FCM: The fuzzy *c*-means clustering algorithm. *Computers & Geosciences*, *10*(2–3), 191–203. http://doi.org/10.1016/0098-3004(84)90020-7

Bird, I., Bos, K., Brook, N., Duellmann, D., Eck, C., Fisk, I., … Wenaus, T. (2005). "LHC computing Grid. Technical design report" CERN-LHCC-2005-024. Retrieved from https://cdsweb.cern.ch/record/840543/files/lhcc-2005-024.pdf

Bjorken, J. D., & Paschos, E. A. (1969). Inelastic Electron-Proton and γ -Proton Scattering and the Structure of the Nucleon. *Physical Review*, *185*(5), 1975–1982. http://doi.org/10.1103/PhysRev.185.1975

Bloom, E. D., Coward, D. H., DeStaebler, H., Drees, J., Miller, G., Mo, L. W., … Kendall, H. W. (1969). High-Energy Inelastic e − p Scattering at 6° and 10°. *Physical Review Letters*, *23*(16), 930–934. http://doi.org/10.1103/PhysRevLett.23.930

Bloom, K. (2015). CMS Software and Computing : Ready for Run 2 arXiv : 1509 . 08180v2 [physics . comp-ph] 30 Sep 2015.

Bloom, K., Boccali, T., Bockelman, B., Bradley, D., Dasu, S., Dost, J., … Zvada, M. (2015). Any Data, Any Time, Anywhere: Global Data Access for Science. Retrieved from http://arxiv.org/abs/1508.01443

Blumenfeld, B., Dykstra, D., Lueking, L., & Wicklund, E. (2008). CMS conditions data access using FroNTier. *Journal of Physics: Conference Series*, *119*(7), 72007. http://doi.org/10.1088/1742-6596/119/7/072007

Boudoul, G., Franzoni, G., Norkus, A., Pol, A., Srimanobhas, P., & Vlimant, J.-R. (2015). Monte Carlo Production Management at CMS. *Journal of Physics: Conference Series*, *664*(7), 72018. http://doi.org/10.1088/1742-6596/664/7/072018

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, *24*(2), 123–140. http://doi.org/10.1023/A:1018054314350

Breiman, L. (2001). Random Forest. *Machine Learning*, *45*(1), 5–32.

http://doi.org/10.1023/A:1010933404324

Buckley, M. R., Lykken, J. D., Rogan, C., & Spiropulu, M. (2014). Super-razor and searches for sleptons and charginos at the LHC. *Physical Review D*, *89*(5), 55020. http://doi.org/10.1103/PhysRevD.89.055020

Buncic, P., Sanchez, C. A., Blomer, J., Franco, L., Harutyunian, A., Mato, P., & Yao, Y. (2010). CernVM – a virtual software appliance for LHC applications. *Journal of Physics: Conference Series*, *219*(4), 42003. http://doi.org/10.1088/1742-6596/219/4/042003

Cahn, R. N., & Goldhaber, G. (2001). *The Experimental Foundations of Particle Physics*. Cambridge: Cambridge University Press. http://doi.org/10.1017/CBO9780511609923

Cha, S. (2007). Comprehensive Survey on Distance / Similarity Measures between Probability Density Functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, *1*(4), 300–307. http://doi.org/10.1007/s00167-009-0884-z

Chacko, Z., Luty, M. A., Nelson, A. E., & Pontón, E. (2000). Gaugino mediated supersymmetry breaking. *Journal of High Energy Physics*, *2000*(1), 003–003. http://doi.org/10.1088/1126-6708/2000/01/003

Cinquilli, M., Spiga, D., Grandi, C., Hernàndez, J. M., Konstantinov, P., Mascheroni, M., … Vaandering, E. (2012). CRAB3: Establishing a new generation of services for distributed analysis at CMS. *Journal of Physics: Conference Series*, *396*(3), 32026. http://doi.org/10.1088/1742-6596/396/3/032026

CMS Collaboration. (n.d.). *The CMS muon project, technical design report, CERN-LHCC-97-032*. Retrieved from http://cdsweb.cern.ch/record/343814

CMS Collaboration. (2008). The CMS experiment at the CERN LHC. *Jinst*, *8004*. http://doi.org/10.1088/1748-0221/3/08/S08004

CMS Collaboration. (2010a). Measurement of the Inclusive Upsilon production cross section in pp collisions at sqrt(s)=7 TeV. http://doi.org/10.1103/PhysRevD.83.112004

CMS Collaboration. (2010b). Performance of the CMS hadron calorimeter with cosmic ray muons and LHC beam data. *Journal of Instrumentation*, *5*(3), T03012–T03012. http://doi.org/10.1088/1748-0221/5/03/T03012

CMS Collaboration. (2011). J/psi and psi(2S) production in pp collisions at sqrt(s) = 7 TeV, *12*. http://doi.org/10.1007/JHEP02(2012)011

CMS Collaboration. (2012a). Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Physics Letters, Section B: Nuclear, Elementary Particle and High-Energy Physics*, *716*(1), 30–61. http://doi.org/10.1016/j.physletb.2012.08.021

CMS Collaboration. (2012b). Performance of CMS muon reconstruction in pp collision events at √s = 7 TeV. *Journal of Instrumentation*, *7*(10), P10002–P10002. http://doi.org/10.1088/1748-0221/7/10/P10002

CMS Collaboration. (2014). Searches for electroweak production of charginos, neutralinos, and sleptons decaying to leptons and W, Z, and Higgs bosons in pp collisions at 8 TeV. *The European Physical Journal C*. http://doi.org/10.1140/epjc/s10052-014-3036-7

CMS Collaboration. (2015). Search for a standard model-like Higgs boson in the uu and ee decay channel at LHC. *Physics Letters B*, *744*, 184–207. http://doi.org/10.1016/j.physletb.2015.03.048

CMS Collaboration. (2016). Search for neutral MSSM Higgs bosons decaying to μ+μ− in pp collisions at. *Physics Letters B*, *752*(1), 221–246. http://doi.org/10.1016/j.physletb.2015.11.042

CMS Collaborations. (2012). Search for a light pseudoscalar Higgs boson in the dimuon decay channel in pp collisions at radicals = 7 TeV. *Phys Rev Lett*, *109*(12), 121801. http://doi.org/10.1103/PhysRevLett.109.121801

CMS Collaborations. (2014). Searches for heavy Higgs bosons in two-Higgs-doublet models and for t → c h decay using multilepton and diphoton final states in p p collisions at 8 TeV. *Physical Review D*, *90*(11), 112013. http://doi.org/10.1103/PhysRevD.90.112013

Craig, N. (2013). The State of Supersymmetry after Run I of the LHC, 72. Retrieved from http://arxiv.org/abs/1309.0528

Cristianini, N., & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge: Cambridge University Press. http://doi.org/10.1017/CBO9780511801389

Cuadros-Vargas, E., Romero, R., & Obermayer, K. (2003). Speeding up Algorithms of the SOM Family for Large and High Dimensional Databases. *Proceedings WSOM*, (i), 167–172. Retrieved from http://www.ni.tu-berlin.de/fileadmin/fg215/articles/cuad_03_wsom.pdf

D0 Collaboration. (2009). Evidence of W W and W Z Production with lepton + jets</mte. *Physical Review Letters*, *102*(16), 161801. http://doi.org/10.1103/PhysRevLett.102.161801

D0 Collaboration. (2011). Search for the Standard Model Higgs Boson in the H → WW → l ν q ′ q ¯ Decay C. *Physical Review Letters*, *106*(17), 171802. http://doi.org/10.1103/PhysRevLett.106.171802

D0 Collaboration. (2012a). Measurements of W W and W Z Production in W + jets Final States in <m. *Physical Review Letters*, *108*(18), 181803. http://doi.org/10.1103/PhysRevLett.108.181803

D0 Collaboration. (2012b). Search for the Standard Model Higgs Boson in Z H → ℓ + ℓ − b b ¯ Product. *Physical Review Letters*, *109*(12), 121803. http://doi.org/10.1103/PhysRevLett.109.121803

D0 Collaboration. (2014). Observation and studies of double J = ψ production at the Tevatron, *111101*, 1–9.

de Alwis, S. P. (2008). Anomaly mediated supersymmetry breaking. *Physical Review D*, *77*(10), 105020. http://doi.org/10.1103/PhysRevD.77.105020

Collaboration. (1991). The DELPHI detector at LEP. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *303*(2), 233–276. http://doi.org/10.1016/0168-9002(91)90793-P

Dittenbach, M., Merkl, D., & Rauber, a. (2000). The growing hierarchical self-organizing map. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, *6*(FEBRUARY 2000). http://doi.org/10.1109/IJCNN.2000.859366

, K.-L., & Swamy, M. N. S. (2014). *Neural Networks and Statistical Learning*. London: Springer London. http://doi.org/10.1007/978-1-4471-5571-3

Dunn, J. C. (1973). A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics*. http://doi.org/10.1080/01969727308546046

Ellert, M., Grønager, M., Konstantinov, A., Kónya, B., Lindemann, J., Livenson, I., … Wäänänen, A. (2007). Advanced Resource Connector middleware for lightweight computational Grids. *Future Generation Computer Systems*, *23*(2), 219–240. http://doi.org/10.1016/j.future.2006.05.008

Evans, D., Fisk, I., Holzman, B., Melo, A., Metson, S., Pordes, R., … Tiradani, A. (2011). Using Amazon's Elastic Compute Cloud to dynamically scale CMS computational resources. *Journal of Physics: Conference Series*, *331*(6), 62031. http://doi.org/10.1088/1742-6596/331/6/062031

Evans, J., Kilminster, B., Luty, M., & Whiteson, D. (2012). Searching For Resonances inside Top-like Events. http://doi.org/10.1103/PhysRevD.85.011104

Fahad, a, Alshatri, N., Tari, Z., Alamri, a, Khalil, I., Zomaya, a, … Bouras, a. (2014). A Survey of Clustering Algorithms for Big Data: Taxonomy & Empirical Analysis. *IEEE Transactions on Emerging Topics in Computing*, *2*(3), 1–1. http://doi.org/10.1109/TETC.2014.2330519

Fajardo, E., Gutsche, O., Foulkes, S., Linacre, J., Spinoso, V., Lahiff, A., … Mohapatra, A. (2012). A new era for central processing and production in CMS. *Journal of Physics: Conference Series*, *396*(4), 42018. http://doi.org/10.1088/1742-6596/396/4/042018

Fajardo, E. M., Dost, J. M., Holzman, B., Tannenbaum, T., Letts, J., Tiradani, A., … Mason, D. (2015). How much higher can HTCondor fly? *Journal of Physics: Conference Series*, *664*(6), 62014. http://doi.org/10.1088/1742-6596/664/6/062014

Falkowski, A., Lee, H. M., & Lüdeling, C. (2005). Gravity mediated supersymmetry breaking in six dimensions. *Journal of High Energy Physics*, *2005*(10), 090–090. http://doi.org/10.1088/1126-6708/2005/10/090

Ferreira Costa, J. A., & de Andrade Netto, M. L. (2001). Clustering of complex shaped data sets via Kohonen maps and mathematical morphology. In B. V. Dasarathy (Ed.), *Data Mining and Knowledge Discovery: Theory, Tools, and Technology III* (pp. 16–27). http://doi.org/10.1117/12.421088

Gan, G., Ma, C., & Wu, J. (2007). *Data Clustering: Theory, Algorithms, and Applications*. *ASASIAM Series on Statistics and Applied Probability* (Vol. 20).

http://doi.org/10.1111/j.1751-5823.2007.00039_2.x

GFAL2 utility tools. (n.d.). Retrieved January 19, 2016, from https://dmc.web.cern.ch/projects/gfal2-utils

Giffels, M., Guo, Y., Kuznetsov, V., Magini, N., & Wildish, T. (2014). The CMS Data Management System. *Journal of Physics: Conference Series*, *513*(4), 42052. http://doi.org/10.1088/1742-6596/513/4/042052

Giudice, G. F., & Rattazzi, R. (1999). Theories with gauge-mediated supersymmetry breaking. *Physics Reports*, *322*(6), 419–499. http://doi.org/10.1016/S0370-1573(99)00042-3

Görür, D., & Edward Rasmussen, C. (2010). Dirichlet Process Gaussian Mixture Models: Choice of the Base Distribution. *Journal of Computer Science and Technology*, *25*(4), 653–664. http://doi.org/10.1007/s11390-010-9355-8

Griffiths, D. (1987). *Introduction to Elementary Particles*. (D. Griffiths, Ed.). Weinheim, Germany: Wiley-VCH Verlag GmbH. http://doi.org/10.1002/9783527618460

Grira, N., Crucianu, M., & Boujemaa, N. (2004). Unsupervised and Semi-supervised Clustering: A Brief Survey. *A Review of Machine Learning Techniques for Processing Multimedia Content, Report of the MUSCLE European Network of Excellence (6th Framework Programme)*, 1–12. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.4074

Gross, K. (2012). LCG Utilities. Retrieved January 19, 2016, from https://twiki.opensciencegrid.org/bin/view/Documentation/Release3/LcgUtilities

Gupta, L., Upadhye, A. M., Denby, B., Amendolia, S. R., & Grieco, G. (1992). Neural network trigger algorithms for heavy quark event selection in a fixed target high energy physics experiment. *Pattern Recognition*, *25*(4), 413–421. http://doi.org/10.1016/0031-3203(92)90089-2

H.E.S.S Collaborations. (2009). Probing the ATIC peak in the cosmic-ray electron spectrum with H.E.S.S. *Astronomy and Astrophysics*, *508*(2), 561–564. http://doi.org/10.1051/0004-6361/200913323

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*.

New York, NY: Springer New York. http://doi.org/10.1007/978-0-387-84858-7

Heikkinen, A., Kaitaniemi, P., Karimäki, V., Kortelainen, M. J., Lampén, T., Lehti, S., … Wendland, L. (2010). Ideal τ tagging with the multivariate data-analysis toolkit TMVA. *Journal of Physics: Conference Series*, *219*(3), 32010. http://doi.org/10.1088/1742-6596/219/3/032010

Ho, T. K., & Bernadó-Mansilla, E. (2006). Data Complexity in Pattern Recognition. In M. Basu & T. K. Ho (Eds.), . London: Springer London. http://doi.org/10.1007/978-1-84628-172-3

Hong, Y., Singh, N., Kwitt, R., & Niethammer, M. (2015). Group Testing for Longitudinal Data (pp. 139–151). http://doi.org/10.1007/978-3-319-19992-4_11

Honkanen, H., Liuti, S., Carnahan, J., Loitiere, Y., & Reynolds, P. R. (2009). New avenue to the parton distribution functions: Self-organizing maps. *Physical Review D - Particles, Fields, Gravitation and Cosmology*, *79*(3). http://doi.org/10.1103/PhysRevD.79.034022

Hufnagel, D. (2015). The CMS TierO goes Cloud and Grid for LHC Run 2. *Journal of Physics: Conference Series*, *664*(3), 32014. http://doi.org/10.1088/1742-6596/664/3/032014

Ingo, S., & Andreas, C. (2008). *Support Vector Machines*. New York, NY: Springer New York. http://doi.org/10.1007/978-0-387-77242-4

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, *31*(8), 651–666. http://doi.org/10.1016/j.patrec.2009.09.011

Jain, A. K., & Dubes, R. C. (1988). Algorithms for Clustering Data. Englewood Cliffs, New Jersey: Prentice Hall.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, *31*(3), 264–323. http://doi.org/10.1145/331499.331504

Kaufman, L., & Rousseeuw, P. J. (1987). Clustering by means of medoids. *Statistical Data Analysis Based on the L 1-Norm and Related Methods. First International Conference*, 405–416416.

Khan, S. S., & Madden, M. G. (2010). A Survey of Recent Trends in One Class Classification (pp. 188–197). http://doi.org/10.1007/978-3-642-17080-5_21

Knuth, D. E. (1974). Computer programming as an art. *Communications of the ACM*,

17(12), 667–673. http://doi.org/10.1145/361604.361612

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*(1), 59–69. http://doi.org/10.1007/BF00337288

Kohonen, T. (2013). Essentials of the self-organizing map. *Neural Networks*, *37*, 52–65. http://doi.org/10.1016/j.neunet.2012.09.018

Koide, Y. (2009). Charged lepton mass spectrum and supersymmetric yukawaon model. *Physics Letters B*, *681*(1), 68–73. http://doi.org/10.1016/j.physletb.2009.09.065

Lange, J. S., & Freiesleben, H. (1996). A parameter-free non-growing self-organizing map based upon gravitational principles: Algorithm and applications (pp. 827–832). http://doi.org/10.1007/3-540-61510-5_139

Lange, J. S., Fukunaga, C., Tanaka, M., & Bozek, A. (1999). Transputer self-organizing map algorithm for beam background rejection at the BELLE silicon vertex detector. *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *420*(1–2), 288–309. http://doi.org/10.1016/S0168-9002(98)00661-5

Lange, J. S., Hermanoski, P., & Freiesleben, H. (1997). A parameter free \mboxself-organizing map for the analysis of pp-reactions at COSY. *Nuclear Instruments and Methods in Physics Research A*, *389*(August), 214–218. http://doi.org/10.1016/S0168-9002(97)00120-4

LHCb Collaborations. (2014). Measurement of Y production in p p collisions at $$ =2.76TeV s = 2.76 TeV. *The European Physical Journal C*, *74*(4), 2835. http://doi.org/10.1140/epjc/s10052-014-2835-1

MAGIC Collabration. (2007). The MAGIC Project: Contributions to ICRC 2007. Retrieved from http://arxiv.org/abs/0709.3763

Marco, C., Fabio, C., Alvise, D., Antonia, G., Francesco, G., Alessandro, M., … Francesco, P. (2009). The gLite Workload Management System (pp. 256–268). http://doi.org/10.1007/978-3-642-01671-4_24

Markou, M., & Singh, S. (2003). Novelty detection: a review—part 1: statistical approaches. *Signal Processing*, *83*(12), 2481–2497. http://doi.org/10.1016/j.sigpro.2003.07.018

Martin, S. P. (1997). A Supersymmetry Primer. Retrieved from http://arxiv.org/abs/hep-ph/9709356

Mascheroni, M., Balcas, J., Belforte, S., Bockelman, B. P., Hernandez, J. M., Ciangottini, D., … Vaandering, E. (2015). CMS distributed data analysis with CRAB3. *Journal of Physics: Conference Series*, *664*(6), 62038. http://doi.org/10.1088/1742-6596/664/6/062038

Massie, M., Li, B., Nicholes, B., Vuksan, V., Alexander, R., Buchbinder, J., … Pocock, D. (2012). *Monitoring with Ganglia Tracking Dynamic Host and Application Metrics at Scale*. O'Reilly Media.

Metson, S., Bonacorsi, D., Ferreira, M. D., & Egeland, R. (2010). SiteDB: Marshalling people and resources available to CMS. *Journal of Physics: Conference Series*, *219*(7), 72044. http://doi.org/10.1088/1742-6596/219/7/072044

Minh, H. Q., Niyogi, P., & Yao, Y. (2006). Mercer's Theorem, Feature Maps, and Smoothing. In *Learning Theory, Lecture Notes in Computer Science* (pp. 154–168). http://doi.org/10.1007/11776420_14

Mirkin, B. (1997). Mathematical Classification and Clustering. *Journal of the Operational Research Society*, *48*(8), 852–852. http://doi.org/10.1057/palgrave.jors.2600836

Narsky, I., & Porter, F. C. (2013). *Statistical Analysis Techniques in Particle Physics*. Weinheim, Germany: Wiley-VCH Verlag GmbH & Co. KGaA. http://doi.org/10.1002/9783527677320

Olive, K. A. (2014). Review of Particle Physics. *Chinese Physics C*, *38*(9), 90001. http://doi.org/10.1088/1674-1137/38/9/090001

Open Grid Scheduler. (n.d.). Retrieved January 23, 2016, from http://gridscheduler.sourceforge.net/

Ovyn, S., Rouby, X., & Lemaitre, V. (2009). Delphes, a framework for fast simulation of a generic collider experiment. Retrieved from http://arxiv.org/abs/0903.2225

Pardo, L. (2005). Divergence Measures. In *Statistical Inference Based on Divergence Measures* (pp. 1–53). Chapman and Hall/CRC. http://doi.org/doi:10.1201/9781420034813.ch1

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2012). Scikit-learn: Machine Learning in Python, *12*, 2825–2830. Retrieved from http://arxiv.org/abs/1201.0490

Pelleg, D., & Moore, A. (1999). Accelerating Exact k-means Algorithms with Geometric Reasoning. In *Proceedings of the Fifth International Conference on Knowledge Discovery in Databases* (pp. 277--281). http://doi.org/10.1145/312129.312248

Pelleg, D., Pelleg, D., Moore, A. W., & Moore, A. W. (2000). X-means: Extending K-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning table of contents* (pp. 727–734). Retrieved from http://portal.acm.org/citation.cfm?id=657808

Perkins, D. H. (1987). *Introduction to High Energy Physics*. Addison-Wesley Publishing Company.

Pordes, R., Petravick, D., Kramer, B., Olson, D., Livny, M., Roy, A., … Quick, R. (2007). The open science grid. *Journal of Physics: Conference Series*, *78*, 12057. http://doi.org/10.1088/1742-6596/78/1/012057

Riahi, H., Wildish, T., Ciangottini, D., Hernández, J. M., Andreeva, J., Balcas, J., … Vaandering, E. W. (2015). AsyncStageOut: Distributed user data management for CMS Analysis. *Journal of Physics: Conference Series*, *664*(6), 62052. http://doi.org/10.1088/1742-6596/664/6/062052

Riordan, M. (1992). The discovery of quarks. *Science (New York, N.Y.)*, *256*(5061), 1287–1293. http://doi.org/10.1126/science.256.5061.1287

Robichaud-Véronneau, A. (2013). Searches for Supersymmetry and Exotics phenomena with the ATLAS detector. *Journal of Physics: Conference Series*, *455*(32), 12012. http://doi.org/10.1088/1742-6596/455/1/012012

Rogan, C. (2010). Kinematical variables towards new dynamics at the LHC. Retrieved from http://arxiv.org/abs/1006.2727

Schölkopf, B., Smola, A., & Müller, K.-R. (1998). Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*. http://doi.org/10.1162/089976698300017467

Seiffert, U., & Jain, L. C. (Eds.). (2002). *Self-Organizing Neural Networks* (Vol. 78). Heidelberg: Physica-Verlag HD. http://doi.org/10.1007/978-3-7908-1810-9

Sfiligoi, I., Bradley, D. C., Holzman, B., Mhashilkar, P., Padhi, S., & Wurthwein, F. (2009). The Pilot Way to Grid Resources Using glideinWMS. In *2009 WRI World Congress on Computer Science and Information Engineering* (pp. 428–432). IEEE. http://doi.org/10.1109/CSIE.2009.950

Singler, J. A. (1996). *Education: Ends and Means (Lynchburg College Symposium Readings) Vol. 9*. University Press of America,.

Sjöstrand, T., Mrenna, S., & Skands, P. (2006). PYTHIA 6.4 physics and manual. *Journal of High Energy Physics*, *2006*(5), 026–026. http://doi.org/10.1088/1126-6708/2006/05/026

Skurichina, M., & Duin, R. P. W. (2002). Bagging, Boosting and the Random Subspace Method for Linear Classifiers. *Pattern Analysis & Applications*, *5*(2), 121–135. http://doi.org/10.1007/s100440200011

Steinbach, M., Karypis, G., & Kumar, V. (2000). A Comparison of Document Clustering Techniques. *KDD Workshop on Text Mining*, *400*, 1–2. http://doi.org/10.1109/ICCCYB.2008.4721382

Ster, D. C. van der, Elmsheuser, J., García, M. Ú., & Paladin, M. (2011). HammerCloud: A Stress Testing System for Distributed Analysis. *Journal of Physics: Conference Series*, *331*(7), 72036. http://doi.org/10.1088/1742-6596/331/7/072036

Thain, D., Tannenbaum, T., & Livny, M. (2005). Distributed computing in practice: the Condor experience. *Concurrency and Computation: Practice and Experience*, *17*(2–4), 323–356. http://doi.org/10.1002/cpe.938

The ATLAS Collaboration. (2012). *Search for the Standard Model Higgs boson produced in association with top quarks in proton-proton collisions at √s = 7 TeV using the ATLAS detector*. *ATLAS-CONF*. Retrieved from https://cds.cern.ch/record/1478423%5Cnhttps://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/CONFNOTES/ATLAS-CONF-2012-135/

Tryba, V., & Goser, K. (1991). A modified algorithm for self-organizing maps based on the Schrödinger equation. In *Artificial Neural Networks* (pp. 33–47). Berlin/Heidelberg: Springer-Verlag. http://doi.org/10.1007/BFb0035875

Uzan, J.-P., & Leclercq, B. (2008). *The Natural Laws of the Universe*. New York, NY: Praxis. http://doi.org/10.1007/978-0-387-74081-2

Vaiciulis, a. (2002). Support Vector Machines in Analysis of Top Quark Production, 8. http://doi.org/10.1016/S0168-9002(03)00479-0

Vannerem, P., Mueller, K.-R., Schoelkopf, B., Smola, A., & Soldner-Rembold, S. (1999). Classifying LEP Data with Support Vector Algorithms, 7. http://doi.org/10.1.1.46.8631

Vesanto, J., & Alhoniemi, E. (2000). Clustering of the self-organizing map. *IEEE Transactions on Neural Networks*, *11*(3), 586–600. http://doi.org/10.1109/72.846731

Villmann, T., Der, R., Herrmann, M., & Martinetz, T. M. (1997). Topology preservation in self-organizing feature maps: exact definition and measurement. *IEEE Transactions on Neural Networks*, *8*(2), 256–266. http://doi.org/10.1109/72.557663

Wan Abdullah, W. A. T. (1992). A Two tier neural network for b tagging. *Fifth Asia-Pacific-Physics Conference*, 435–438.

Wang, F., & Sun, J. (2015). Survey on distance metric learning and dimensionality reduction in data mining. *Data Mining and Knowledge Discovery*, *29*(2), 534–564. http://doi.org/10.1007/s10618-014-0356-z

Whiteson, S., & Whiteson, D. (2009). Machine learning for event selection in high energy physics. *Engineering Applications of Artificial Intelligence*, *22*(8), 1203–1217. http://doi.org/10.1016/j.engappai.2009.05.004

Wildish, T. (2015). Understanding the T2 traffic in CMS during Run-1. *Journal of Physics: Conference Series*, *664*(3), 32034. http://doi.org/10.1088/1742-6596/664/3/032034

Wu, Y., & Takatsuka, M. (2005). The Geodesic Self-Organizing Map and its error analysis. *Conferences in Research and Practice in Information Technology Series*, *38*, 343–352.

Wu, Y., & Takatsuka, M. (2006). Spherical self-organizing map using efficient indexed geodesic data structure. *Neural Networks*, *19*(6–7), 900–910. http://doi.org/10.1016/j.neunet.2006.05.021

Yu, H. (2003). SVMC: Single-class classification with support vector machines. *IJCAI*

International Joint Conference on Artificial Intelligence, 567–572.

ZEUS Collaboration. (1993). The ZEUS Detector. Retrieved January 15, 2016, from http://www-zeus.desy.de/bluebook/bluebook.html

LIST OF PUBLICATIONS

CMS collaboration. Search for diphoton resonances in the mass range from 150 to 850 GeV in pp collisions at. *Physics letters B* 750 (2015): 494-519.

CMS collaboration. Search for neutral MSSM Higgs bosons decaying to μ+ μ− in pp collisions at. *Physics letters B* 752 (2016): 221-246.

Mascheroni, M., Balcas, J., Belforte, S., Bockelman, B. P., Hernandez, J. M., Ciangottini, D., … Vaandering, E. (2015). CMS distributed data analysis with CRAB3. *Journal of Physics: Conference Series*, *664*(6), 62038. http://doi.org/10.1088/1742-6596/664/6/062038